University of the Philippines, Manila

College of Arts and Sciences

Department of Physical Sciences

GEEBBO:
Generic, Easy-to-manage Expert system Builder for Business Organizations (Using Drools)

A Special Problem in partial fulfillment
of the requirements for the
Degree of Bachelor of Science in Computer Science

Submitted by:

Genesie Jo A. Del Rosario

April 2010

Abstract

Expert systems are very helpful in solving problems quickly and efficiently within a specialized domain. From simple decision making to medical diagnosis, the applications of expert systems now also include business processes. But because the business world is ever changing, business experts may find it difficult to manage and update their business rules. Hence the need for a user-friendly business rules management system that can be adapted to by different types of business organizations, hence the development of GEEBBO. With the help of Drools, GEEBBO (Generic, Easy-to-manage, Expert system Builder for Business Organizations) allows business experts and organizations to easily manage their business rules. GEEBBO provides a simple way for developers to create a web-based expert system. The *system administrator* manages the settings and the users of the system. The *knowledge engineer or programmer* handles the objects that will be used by the *author*, who, in turn, writes or encodes the rules given by the expert. The *end-users* can choose any available ruleset, supply the needed facts or data, and view the results produced by the rule engine.

Keywords: Expert System, Business Rules Management System, Business Rules

ACCEPTANCE SHEET

The Special Problem entitled "Generic, Easy-to-manage Expert system Builder for Business Organizations (GEEBBO)" prepared and submitted by Genesie Jo A. Del Rosario in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

_____
Avegail D. Carpio, M.S.
Adviser

EXAMINERS:

	Approved	**Disapproved**
1. Gregorio B. Baes, Ph.D. (candidate)	_____	_____
2. Richard Bryann L. Chua, M.S.	_____	_____
3. Aldrich Colin K. Co, M.S. (candidate)	_____	_____
4. Ma. Sheila A. Magboo, M.S.	_____	_____
5. Vincent Peter C. Magboo, M.D., M.S.	_____	_____
6. Geoffrey A. Solano, M.S.	_____	_____
7. Bernie B. Terrado, M.S. (candidate)	_____	_____

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

_____
Geoffrey A. Solano, M.S.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences
and Mathematics

_____
Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences
and Mathematics

_____
Reynaldo H. Imperial, Ph.D.
Dean
College of Arts and Sciences

Acceptance Sheet

Abstract

Table of Contents

I. INTRODUCTION

A. Background of the Study

Expert systems are very helpful in solving problems quickly and efficiently within a specialized domain [1]. One of the major advantages of using expert systems is the increased consistency and standardization of rules. Same results will be obtained even if there are different users, so long as the inputs or data provided are the same. Moreover, procedures for the expert system can be updated to allow for improvements and refinements in the application [2].

Applications of expert systems have now extended to various fields such as medical diagnosis, engineering, accounting and social sciences to mention a few [1]. Expert systems are now widely used in the business world. These expert systems were used as rule based programming tools which later on became rule engines. These rule engines are the core component of what is called Business Rule Management Systems or BRMS [3].

BRMS are responsible for the storage and maintenance of business rules, automation of business processes, making inferences within the rules and allowing business analysts and other non-technical users to create, understand and maintain the rules and policies of the business. BRMS are typically used in automating procedures (like loan application and credit approval), advice giving and decision support (benefits eligibility), planning and scheduling (of budgets, meetings, advertising), diagnosis and detection (of medical conditions, valid and invalid data), classification (of customers, products, risks) and matching and recommending (strategies to investors, products to clients) [4].

Some of the business organizations that make use of BRMS are those related to financial services (mortgage, lending, credit cards), insurance, telecommunications, healthcare, government, and retails (online, mail order), including large-scale national and multinational companies with multiple legacy systems [5]

Currently, there are a number of commercially available BRMS in the market. One of which is the FICO™ Blaze Advisor®. It allows IT and non-technical business experts to work together and easily create, maintain and control business applications quickly and effectively. It also features trees, tables and scorecards for defining and updating rule processes [6]. It offers

a variety of tools and views for design, analysis and debugging. However, non-technical users like normal business analysts would find it difficult to use because the rule language syntax are far from natural language [7].

Another leading BRMS in the market is IBM's ILOG. It provides easy, safe and predictable control over automated business decisions. It offers tools for business managers, analysts, architects and developers. ILOG runs in Java, .NET, mainframe and SOA-based environments [g]. ILOG rules can be written using several languages, from a syntax close to natural language down to a Java-like syntax for developers. But the modeling or customization of the business rules is highly technical, thus, requiring considerable technical skills at an IT professional level [7].

However, these high-end BRMS are extremely expensive. Aside from installation fees, the annual maintenance, runtime fees and professional services would cost up to half a million dollars or more. Hence comes the role of the open source BRMS [8]. There are the OpenRules® and JBoss Drools (Dynamic Rule Object-Oriented Language System), for example.

OpenRules® provides software for Rules-based Application Development. It has a variety of tools and services that gives business analysts the ability to work together with software developers to create and maintain Rules-based applications with complex business and presentation logic [9].

Rule authoring in OpenRules are done via MS Excel, Google Spreadsheets, and Eclipse IDE [9].

Rules void processingLogic(Game g)					
C1	A1	A2	A3	A4	A5
dialog().isCurrent(step)	if (dialog().isInitial()) { initialize(g); dialog().next = step; }	if (g.guess == g.answer) dialog().next = step;	if (g.guess != g.answer) dialog().next = step;	if (g.tryAgain) { dialog().next = step; if (g.guess == g.answer) initialize(g); }	if (g.tryAgain == false) { dialog().next = step; }
String step	String step	String step	String step	String step	String step
Current Step \ Event	**Always**	**Correct Answer**	**Incorrect Answer**	**Try Again = Yes**	**Try Again = No**
	MakeGuess				
MakeGuess	{ g.attempts++; }	Match	NoMatch		
Match				MakeGuess	Goodbye
NoMatch				MakeGuess	Goodbye

Fig. 1 – MS Excel

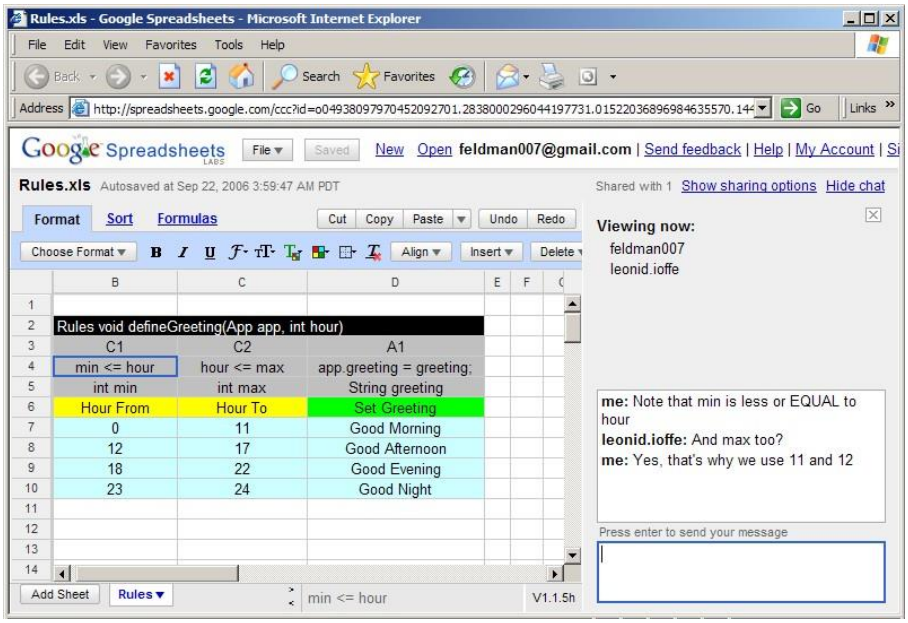

Fig. 2 – Google Spreadsheet

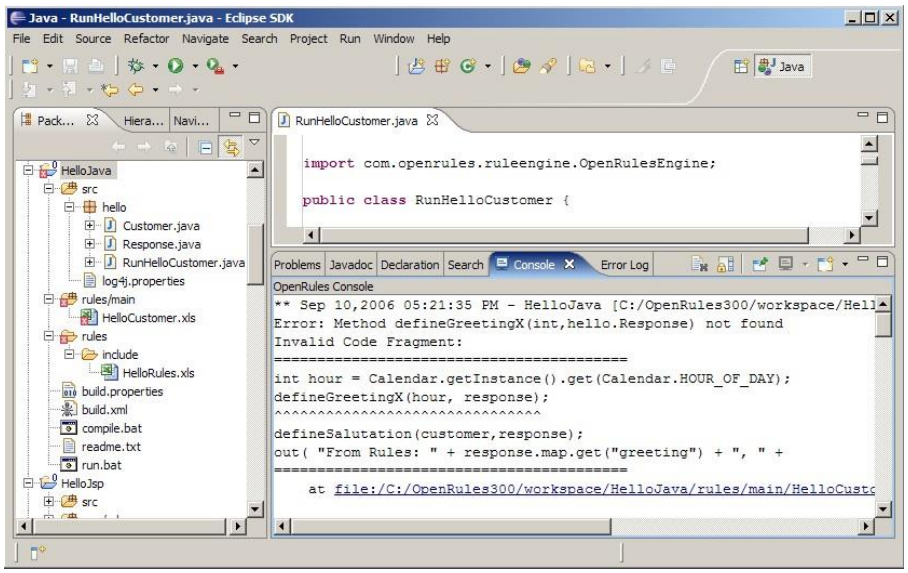

Fig. 3 – Java Eclipse IDE

Drools 5.0 is an open source object-oriented rule engine for Java [8]. One of its sub projects is Drools Guvnor, which is a web-based governance system, traditionally known as BRMS. Drools Guvnor offers web based GUIs, editors and tools that help the users to manage the rules [10].

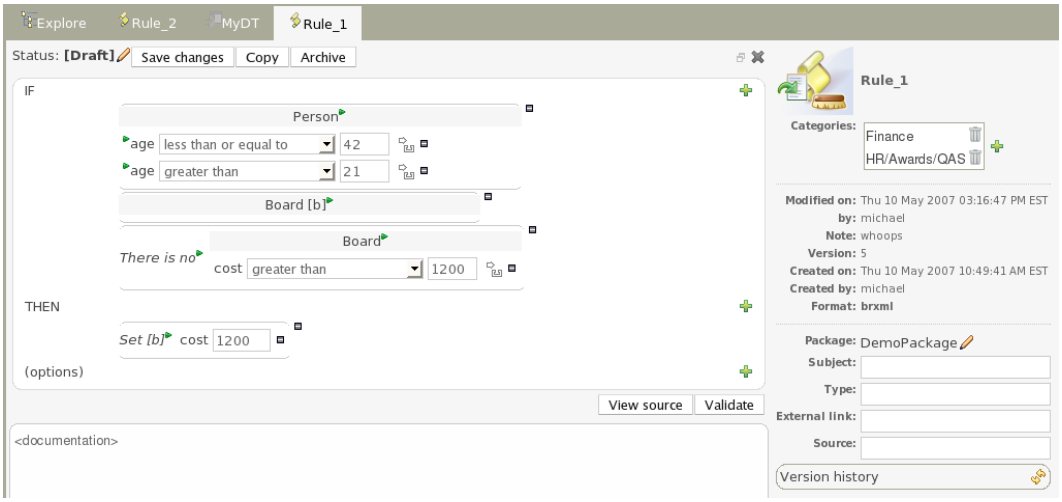

Fig. 4 – Drools Guvnor Graphical Editor

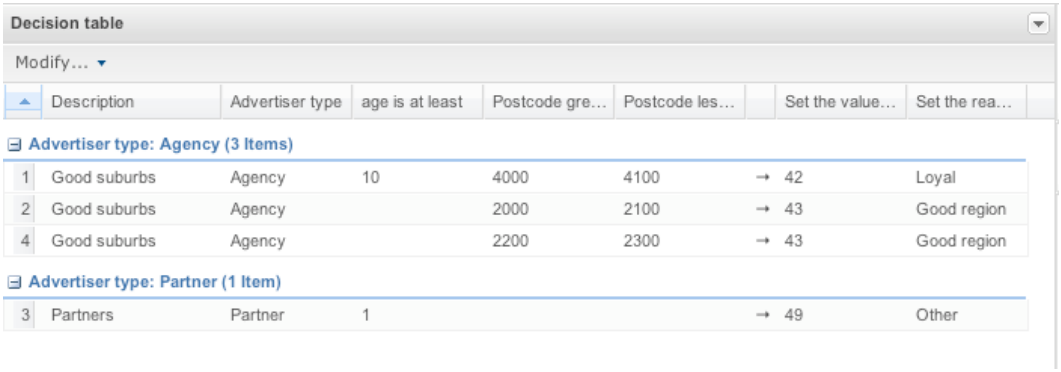

Fig. 5 – Drools Guvnor Decision Table

As seen from the screenshots, both BRMS offers a “user-friendly” interface but still there are some aspects that look and feel technical for ordinary users. There are pieces of Java code that the users may not comprehend at first encounter. Thus, workshops and trainings are being offered to users, specifically to business analysts, for them to be able to effectively use these tools and manage their rules.

B. Statement of the Problem

Business organizations are affected by many factors, like the market economy, stocks exchange and organizational policies. These factors cause inevitable changes which business organizations must accommodate and adapt to [11]. In order to do this, there is a strong need for business to be more agile [12].

However, Expert Systems are generally designed as “closed systems”; they are developed only for a specific application domain. They are built with predefined rules for accomplishing a particular task [13]. Hence, the users of the system, especially business analysts, who are not inclined in the IT field, will find it very difficult to efficiently manage an Expert System on their own.

C. Objective of the Study

To extend Drools to provide a tool for building a web-enabled knowledge-based Expert System for business organizations that provides an easy-to-manage interface with the following capabilities for the following users:

I. System Administrator
   A. Log In/Log out
   B. Update user accounts (Add/Edit/Delete)
   C. Activate/Deactivate user accounts
   D. Manage site and database settings

II. Author – the business expert or the writer of rules
   A. Log In/Log out
   B. Manage rulesets (Add/Edit/Delete)
   C. Manage data source
   D. Manage rulegroup

III. Knowledge Engineer/Programmer
   A. Log in/Log out
   B. Manage objects (Add/Edit/Delete)

IV. End-user

A. Choose a ruleset

B. Input data (values for the object's attributes)

D. Significance of the Study

The Expert System builder is helpful in lessening the amount of time and effort spent in programming or developing a BRMS. By using an object-oriented approach and providing a generic framework for the Expert System, the BRMS is flexible enough to be used in many different application domains.

The dynamic and user-friendly interface featured in the system allows developers to create a BRMS without the need to learn a new language or nomenclature. Moreover, it enables users, even those who are non-technical at all, to easily manage the system according to their needs and requirements, without the actual programming process. Business analysts are able to have full control of their business logic, rather than having to seek IT assistance. In addition to that, the system is freeware, thus, it is very beneficial especially to small companies without regular IT department.

E. Scope and Limitations

1. The system is most suited to be used by business organizations.
2. The type of rules and operations supported by the system is based on the capabilities of the rule engine.
3. The accuracy of the results is dependent on: (1) the clarity and complexity of the rules provided by the author; and (2) rule engine.
4. There is only one expert per domain

F. Assumptions

1. An Expert defines the rules needed for a specific business process.
2. The Knowledge Engineer first creates objects based on the given rules.
3. The Author then encodes the rules provided by the Expert.

4. The Business Client, Knowledge Engineer and End-user provide correct data to the rule engine.
5. There are no conflicting rules.
6. In case of conflicting rules, priority numbers are used to choose the rule to fire, otherwise, both rules will fire.

II. REVIEW OF RELATED LITERATURE

There are several generic categories of knowledge-based expert system applications that emerged in the last years. Those relevant to avionics include avionics equipments design, fault diagnosis, pilot associate system, etc. Airborne equipment design is critical and complex that is why it demands the use of Expert System technology. But aircrafts use different equipments for different purposes, and developing an independent expert system for each equipment is not practical, thus, a *Generic Component Based Expert System for Airborne Equipment Design* was developed [14].

The traditional software program (or Expert System) is decomposed into generic, independent primary elements (component) like Facts Interpreter, Knowledge Base Constructor, Inference Engine, User Interface, etc. These components can be adopted and used to be able to build a new software program more easily [14].

The development of a generic knowledge-based expert system as an incremental learning system aims to analyze the common parameters given to the application domains and model the common behavior of these parameters. Rule-based and similarity-based reasoning are used in developing the system and making generalizations the can make the system adapt more flexible with the environment changes. In effect, the generic expert system is expected to function to define or create another instance of domain-specialist expert systems [15].

Since generic expert systems can be used in many application domains, they must be domain independent. In order to ensure this, an intelligent user interface called *GESIA (Generic Expert System Intelligent Assistant)* was developed. It handles the interaction between the user and the expert system using uncertainty-based reasoning. More than providing a pleasing, easy to use work environment, it aims to recognize the system's application domain and the user's needs to be an intelligent assistant. It collects metrics from the system. The intelligent interface may offer the user a customized interface based from the collected data, or independently make the changes based on perceived user behavior [16].

Business rules are essential parts of a business system model. At present, there are numerous approaches to, definitions and classifications of business rules. Similarly, there are also several approaches in business rules formalization and implementation. Thus, there is a

need for a formal business rules representation that will allow the use of some formal language, yet, still has easy to understand domain model. The paper compares two approaches for that, namely, ontology modeling and UML, each with concrete examples in Protege and ArgoUML, respectively [17].

Both approaches have class as their starting entity which can have slots and attributes. Protégé supports concrete instances, whereas, ArgoUML has methods and operations for constraint definitions. Each tool has their own built-in predicates and operations for complex business rules [17].

It is not easy to tell which the better tool is, but there are factors to consider in choosing between the two. Some of the things to consider are if declarative knowledge is divided from procedural, processes can be included, rules are explicit, queries are enabled, rule consistency can be checked, etc [17].

Business rules and business processes are important in defining the requirements of a software system. Traditionally, rules are scattered inside application code, making it very difficult to manage. Since rules change quickly, it is desirable to externalize the rules and move them outside the application. To be able to effectively manage business rules, there are several business rules approaches that can be used. The paper evaluates three well-known business rules approaches: Von Halle's, Morgan's and Bajec and Krisper's approaches [18].

The Von Halle Methodology deals with rules discovery, built upon four basic principles: separate (rules are separated from the system), trace (from every rule, there is a two-way connection), externalize (rule is accessible to non-technical audience), and position for change (rules are expected to change) [18].

Morgan's Approach is concerned about rules translation in the software system. According to the approach, a simple, unambiguous language can be used to express rules, and that the language is accessible to all users: the business owner, business analyst, technical architect, etc [18].

Bajec Krisper methodology is about defining a business rules model, which will be divided into two sections: one for the Business Level (rule efficiency, impact, history,

documentation) and another for Information System Level (rule description, category, implementation component) [18].

In general, small to midsized businesses do not have sufficient budgets for software licensing and maintenance of BRMS. A cheaper solution that can be used by small and midsized companies for managing their knowledge and support decision making processes is *ARulesXl* rulesets. They are developed and maintained in Microsoft Excel spreadsheets, where the cells are used to provide the inputs and to hold the values of the returned results. The rulesets in the spreadsheet can be incorporated in a variety of languages and tools and can be called form Web applications written in .NET or Java. The spreadsheet was used with the assumption that almost all business analysts and workers have at least minimal knowledge about office application. Moreover, spreadsheet will allow reusing and collecting rules for the company [19].

Cooperation among companies is significant to enhance performance and assure competitive advantages for the business partners involved. To achieve business cooperation, there is a need for scalable, distributed and portable collaborative systems. The article presents a collaborative software application architecture based on Business Process Modeling Notation standard and automated semantic web service coupling for modeling business. In the system, the partners' performance and economic activity influence others. This was done through semantic models and business ontologies [20].

With the fast development of technology, the article presents digital prospects for business management. With the idea that business rules are implement as standards into databases, and that their execution and control are automatic, mobile digital management are considered to be used to manage the business organization [21].

Mobile digital management is defined through partially or totally achieving of at least one of the management functions, forecasting-planning, organization, decision making and command, coordination, motivation, evaluation and control, by using wireless network technology and mobile devices such as PDA, intelligent phones, iPhones, GPS devices or solid hard notebooks. These technologies can then become the fundamental platforms to be used by executive managers for business coordination, evaluation and control [21].

A multi-perspective expert system approach was created based on algebraic sum method. The approach aims to deal with uncertain knowledge and provide different perspectives for different experts while dealing with single knowledge base according to experts' interests. The assertion value or the importance of information to the current query can be evaluated where the algebraic sum would determine the decision in a given perspective [22].

Business Process Management System (BPMS) is the information technology that supports the implementation of the business process management approach. Its main components are a repository for process definition, a repository for process instance, a transaction manager, a connector framework, a process engine and middleware [23].

III. THEORETICAL FRAMEWORK

Expert System

An expert system is a computer program that makes use of artificial intelligence to solve problems within a specialized domain that ordinarily requires human expertise. From its first application in analyzing chemical compounds, its uses now extend up to medical diagnosis, petroleum engineering and financial investing [1].

The two main components of an expert system are the knowledge base and the inference engine. A knowledge base is an organized collection of facts about the system's domain. These facts are obtained from human experts. This knowledge is then represented as "if-then" rules or production rules. An inference engine interprets and evaluates the facts in the knowledge base, together with the data given by the user, in order to arrive at a particular answer or conclusion [1].

Business Rule Management System (BRMS)

A Business Rule Management System (BRMS) is a software system used to define, deploy, execute, monitor and maintain decision logic, also called business rules, that an organization or an enterprise uses [*12]. It helps companies and business experts to easily develop and update business applications, and to make smart business decisions. It allows a fast and accurate implementation of new business strategies and rules [4].

The organizations that benefit most from BRMS are usually the larger organizations like those dealing with financial services (loans, mortgage and credit cards), insurance (health, life and car), healthcare, and telecommunications. [4]

The main components of a BRMS are [*12]:

1. a repository for the rule base or knowledge base
2. a development environment where technical developers and business experts can define and manage business rules
3. a runtime environment where applications can invoke and run business rules using a business rules engine

Business Rules

A business rule is a statement that defines or constrains one or more aspects of a business. It is intended to assert business structure or control or influence the behavior of the business [24]. Business rules usually include policies, requirements and conditional statements needed to determine the actions that must take place in applications and systems [4].

An example of business rules may be about access control issues, such as professors accessing the grades of his/her students only. Business rules may also be about organization policies, like a university policy to expel students who fails more than two courses in a semester. Business rules may also focus on business calculations, like the conversion of a percentage mark to a letter grade [25].

Business Rules Engine or Rule Engine

A business rules engine is a software system that executes one or more business rules in a runtime production environment [26]. A rule engine can be viewed as a sophisticated interpreter of rules, which consist of “if-then” statements. The rule engine is able to determine when to evaluate each rule depending on the input required for the rule, as well as the results from the evaluation of previous rules [28].

DROOLS: Business Logic Integration Platform

Drools is an open source object-oriented rule engine for Java. It is a community release from JBoss.org [27].

Drools 5.0 launches the Business Logic integration Platform (BLiP), which provides a unified and integrated platform for rules, workflow, and event processing [28].

Drools 5.0 is divided into 4 main sub projects [28]:

1. *Drools Guvnor* – a web based governance system (traditionally referred as BRMS) which allows the technical user to create, modify and test rules
2. *Drools Expert* – the traditional rule engine (will be discussed further later on)
3. *Drools Flow* – powerful and extensible workflow engine where users can specify business logic and view the workflow graphically
4. *Drools Fusion* – the event processing side that handles temporal reasoning

Drools Expert [29]

Drools is a Rule Engine that uses rule-based approach in order to implement an Expert System. However, it is more correctly classified as a Production Rules System.

A Productions Rule System deals with knowledge representation in order to provide propositional, first order logic in a clear and concise manner. The Inference Engine is the brain of a Production Rules System. It can handle a large number of rules and facts then match them against Production Rules in order to make conclusions and trigger actions. A Production Rule is a two-part structure using First Order Logic for reasoning over knowledge representation. It is of the form:

```
when
 <conditions>
then
 <actions>;
```

Pattern Matching is the process of matching new or existing facts against Production Rules. The algorithm used by Drools is Rete's algorithm. Drools enhanced and optimized the implementation of the algorithm for object oriented systems and named their Rete implementation ReteOO.

Rule Format [29]

1. **Rule File**

   Drools accepts and process rules through a Rule File, which is a file with .drl extension. The basic parts of a Drools Rule Language (DRL) file are:

   a. `package` *`package-name`* – to declare a package, which is a collection of rules and other related constructs, such as imports and globals, which are related to each other

   b. *`imports`* – to import classes from the Java package for the objects that will be used in the rule

   c. *`globals`* – to define global variables that will be available to the rules. They are typically used to return data from rules, like logs or values added in rule consequences (e.g. scoring)

   d. f*`unctions`* – for creating functions in the rule in order to keep the logic all in one place and invoking actions on the consequence (`then`) part of the rule.

   e. *`rules`* – production rules

   A Rule File may have multiple rules, queries and functions.

2. **Rule Syntax**

```
rule "<name>"
    <attribute>*
when
    <conditional element>*
then
    <action>*
end
```

3. **Rule Attributes**

   Some of the attributes that can be used for the rules:

   a. `salience` – (integer) priority number where the higher integer gets higher priority

   b. `agenda-group` – (String) to allow to fire only the rules in the agenda group that has acquired the focus

4. **Conditional Elements**

   Some of the operators for LHS are:

   a. < - less than

   b. <= - less than or equal

   c. > - greater than

   d. >= - greater than or equal

   e. == - equal

   f. != - not equal

   g. `contains` – checks a field of some Collection type for a value

   h. `not contains` – checks a field of some Collection type for a value

5. **Actions**

   The common actions on RHS are:

   a. `update(object, handle)` – to tell the engine the object has changed and that rules may need to be reconsidered

   b. `modify( <fact-expression> ) {`

      `<expression> ], <expression>]*`

      } – updates facts through setter functions

III. Design and Implementation

A. Entity Relationship Diagram

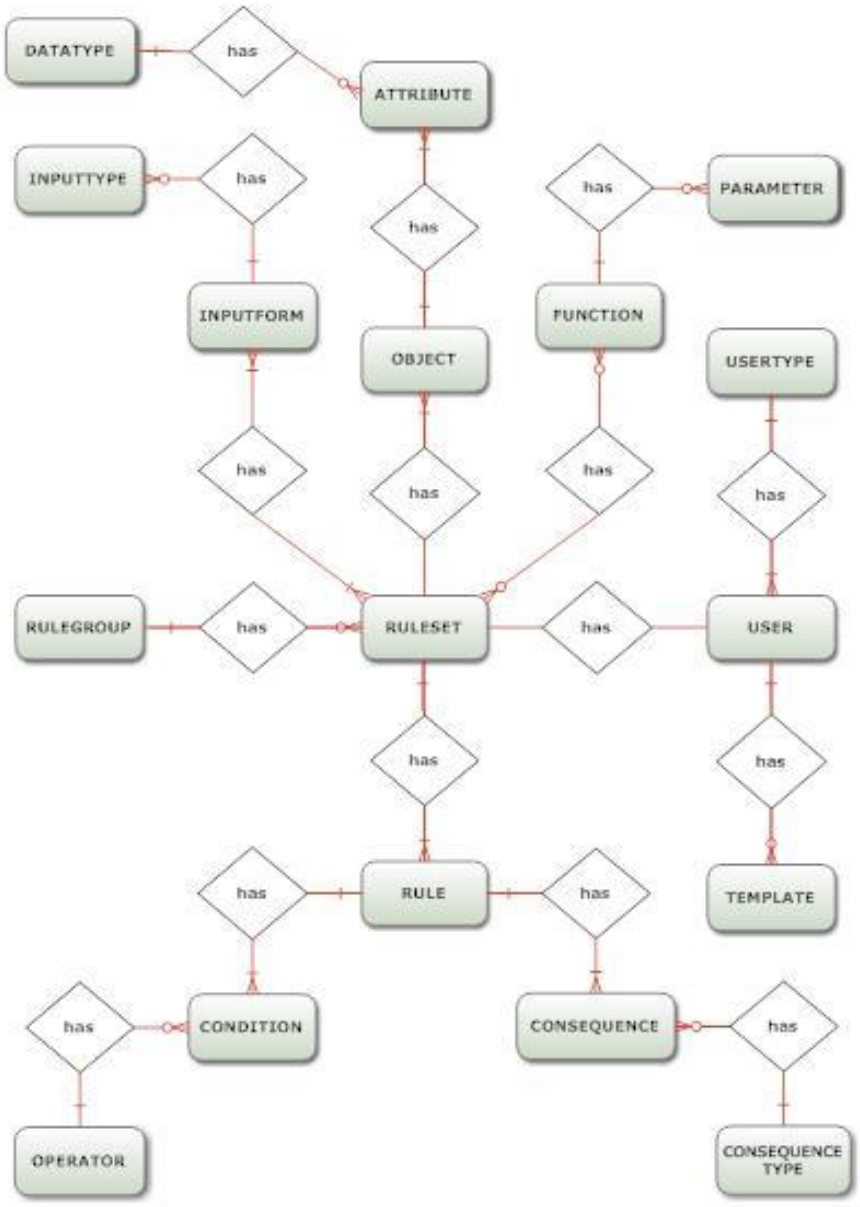

Fig. 6 – Entity Relationship Diagram for GEEBBO

B. Data Dictionary

attribute – attributes of the object

Field	Type	Null	Default	Comments
id	int(11)	No		Unique id of the attribute
objId	int(11)	No		Id of the object to which the attribute belongs
datatype	int(11)	No		Id of the data type of the attribute
variable	varchar(20)	No		Variable name for the class file
name	varchar(30)	No		Name or label for the variable
description	text	No		Description for the attribute

condition – rule condition

Field	Type	Null	Default	Comments
id	int(11)	No		Unique id for the condition
ruleId	int(11)	No		Id of the rule to which the condition belongs
objId	int(11)	Yes	*NULL*	Id of the object used in the condition
attrId	int(11)	Yes	*NULL*	Id of the attribute used in the condition
function	varchar(30)	Yes	*NULL*	Name of the function used in the condition
optrId	int(11)	No		Id of the operator
value	varchar(30)	No		Value being compared to
bool	tinyint(1)	No		0 – or, 1 - and
type	tinyint(1)	No		Condition type (1-4)
seqNo	int(11)	No		Order of condition in the rule

consequence – rule consequence

Field	Type	Null	Default	Comments
id	int(11)	No		Unique id for the consequence
ruleId	int(11)	No		Id of the rule to which the consequence belongs
type	tinyint(1)	No		Consequence type (1-4)
value	varchar(50)	No		Decision or value to assign to a variable
objId	int(11)	No		Id of the object used in the consequence
attrId	int(11)	No		Id of the attribute used in the consequence
function	varchar(30)	Yes	*NULL*	Name of the function used in the consequence
seqNo	int(11)	No		Order of consequence in the rule

consequencetype – type of consequence

Field	Type	Null	Default	Comments
id	tinyint(1)	No		Unique id of the consequence type
type	varchar(15)	No		Type of consequence

datatype - type of data or variable

Field	Type	Null	Default	Comments
id	int(11)	No		Unique id of the data type
type	varchar(10)	No		Type of data

function – ruleset function

Field	Type	Null	Default	Comments
id	int(11)	No		Unique id of the function
rulesetId	int(11)	No		Id of the ruleset to which the function belongs
returntype	int(11)	No		Id of the datatype of the return value
name	varchar(30)	No		Name of the function
code	text	No		Code snippet in Java language
description	varchar(100)	No		Description for the function

inputform – An entry in the input form for a ruleset

Field	Type	Null	Default	Comments
rulesetId	int(11)	No		Id of the ruleset that uses the input form
attrId	int(11)	No		Id of the attribute used
objId	int(11)	No		Id of the object used
question	varchar(100)	No		Question for asking the value of the attribute
inputtype	int(11)	No		Id of the input type
choices	text	Yes	*NULL*	CSV for predefined answers
status	tinyint(1)	No	1	1=the value is required, 0=no need to ask

inputtype – type of form for input

Field	Type	Null	Default	Comments
id	int(11)	No		Unique id of the input type
type	varchar(20)	No		Type of input form

object – object used by rulesets

Field	Type	Null	Default	Comments
id	int(11)	No		Unique id of the object
name	varchar(20)	No		Name of the object
description	text	No		Description of the object
created_by	varchar(20)	No		Username of the creator
date_created	varchar(16)	No		Date created (YYYY-MM-DD HH:MM)
modified_by	varchar(20)	Yes	*NULL*	Username of the recent modifier
date_modified	varchar(16)	Yes	*NULL*	Date modified

operator – operators used for comparisons

Field	Type	Null	Default	Comments
id	int(11)	No		Unique id of operator
operator	varchar(15)	No		Operator symbol
name	varchar(25)	No		Operator name/label

Parameter – parameter fields in functions

Field	Type	Null	Default	Comments
id	int(11)	No		Unique id for the parameter
functionId	int(11)	No		Id of the function that uses the parameter
objId	int(11)	No		Id of the object
attrId	int(11)	No		Id of the attribute

Rule – rule entry in the ruleset

Field	Type	Null	Default	Comments
id	int(11)	No		Unique id of the rule
ruleId	int(11)	No		Order of the rule in the ruleset
rulesetId	int(11)	No		Id of the ruleset
salience	int(11)	No		Priority number
no_loop	tinyint(1)	No		0=false, 1=true
agenda	int(11)	No	0	Agenda or group number
start	tinyint(1)	No		0=false, 1=true

Rulegroup – grouping for ruleset

Field	Type	Null	Default	Comments
id	int(11)	No		Unique id of the rule group
name	varchar(30)	No		Name of the group
description	text	Yes	*NULL*	Description about the group
created_by	varchar(30)	No		Username of the creator
date_created	varchar(16)	No		Date created

Ruleset

Field	Type	Null	Default	Comments
id	int(11)	No		Unique id of the ruleset
name	varchar(50)	No		Ruleset name
description	text	No		Description about the ruleset
type	tinyint(1)	No		1=nonsequential, 2=sequential
objects	varchar(50)	No		CSV of object ids
attributes	varchar(100)	No		CSV of attribute ids
formset	tinyint(1)	No	0	0=false, 1=true
status	tinyint(1)	No	0	0=unpublished, 1=published
created_by	varchar(20)	No		Username of the creator
date_created	varchar(16)	No		Date created
modified_by	varchar(20)	Yes	*NULL*	Username of the recent modifier
date_modified	varchar(16)	Yes	*NULL*	Data last modified
rulegroup	int(11)	No		Id of rulegroup
conclusion	text	No		Default message

Template – template/css for the system

Field	Type	Null	Default	Comments
id	int(11)	No		Unique id of the template
name	varchar(50)	No		Name of the template
username	varchar(30)	No		Username of the uploader
date_uploaded	varchar(16)	No		Date uploaded

Users – user of the system

Field	Type	Null	Default	Comments
username	varchar(10)	No		Unique username
password	varchar(10)	No		Password
name	varchar(50)	No		Complete name of the user
details	text	No		Other details about the user
type	tinyint(1)	No		Id of the usertype of the user
status	tinyint(1)	No		0=deactivated, 1=activated
question	text	Yes	*NULL*	Security question
answer	text	Yes	*NULL*	Answer for security question

Usertype

Field	Type	Null	Default	Comments
id	tinyint(1)	No		Unique id of the usertype
type	varchar(10)	No		Type of user

C. Context Diagram

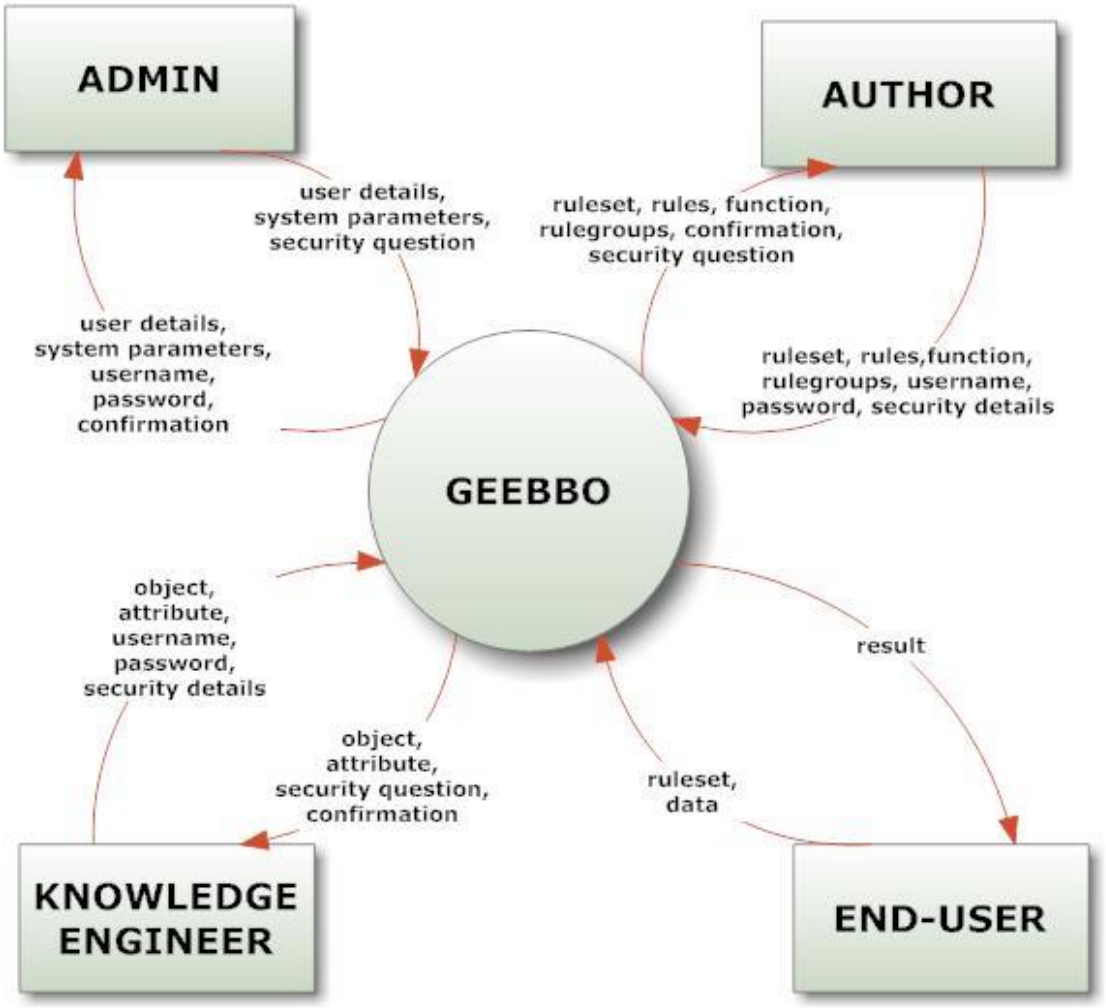

Fig. 7 - Context Diagram for GEEBBO

The four main users of the system are the system administrator, the knowledge engineer or programmer, the author and the end-user. The system administrator manages user accounts and the system configurations. The knowledge engineer or programmer creates objects through Java classes. The author writes or encodes the rules given by the Expert. The end-user gives data to the rule engine.

D. Data Flow Diagram

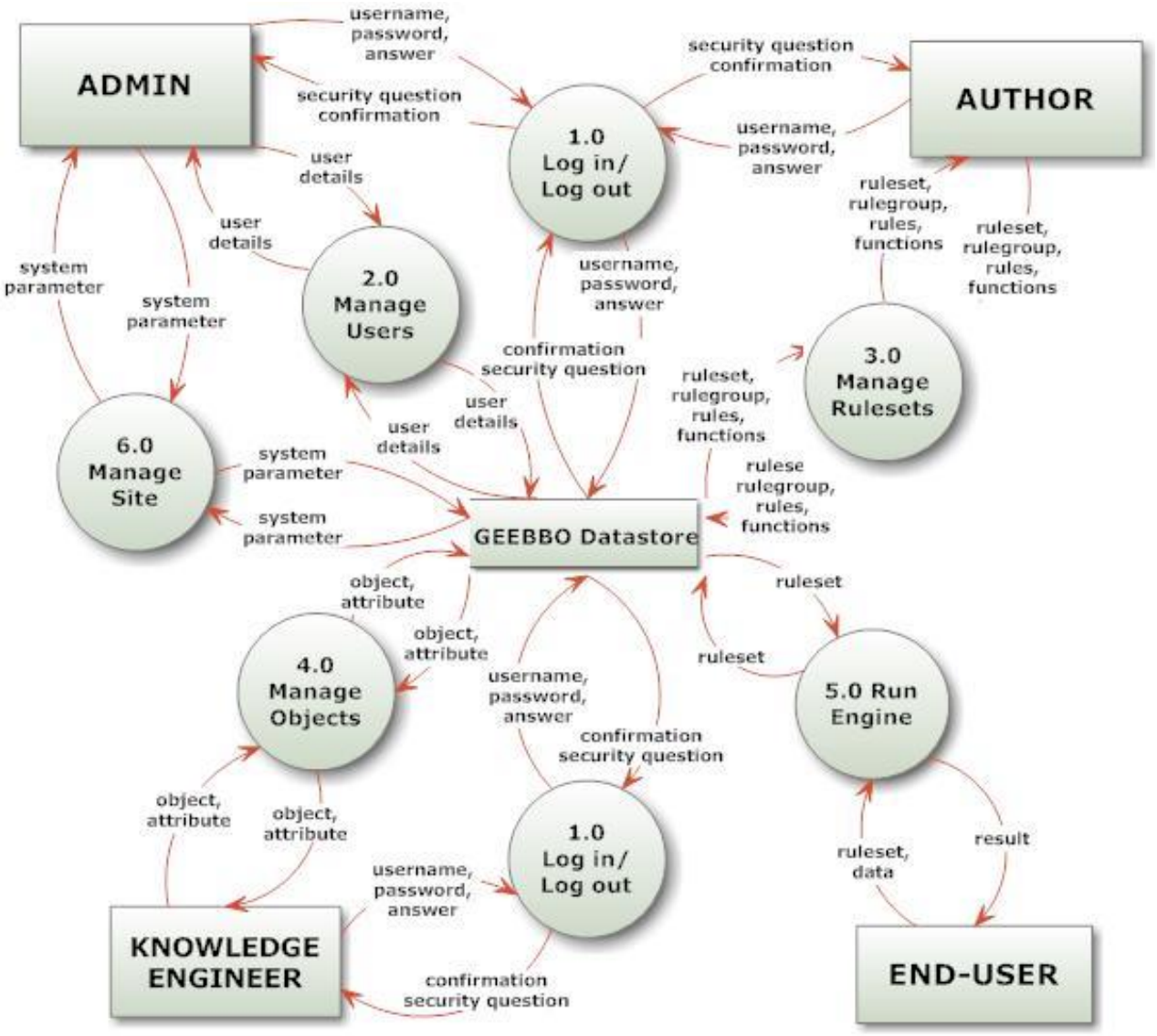

Fig. 8 - Top-Level Diagram for GEEBO

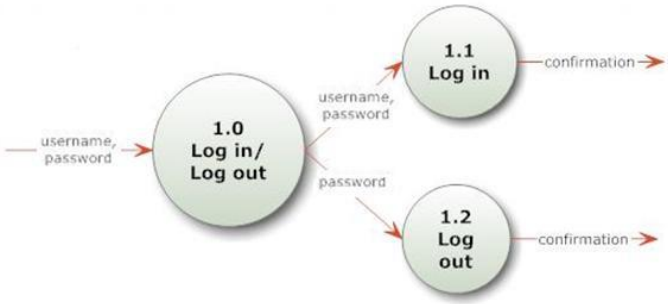

Fig. 9 - Subexplosion for Log in/Log out

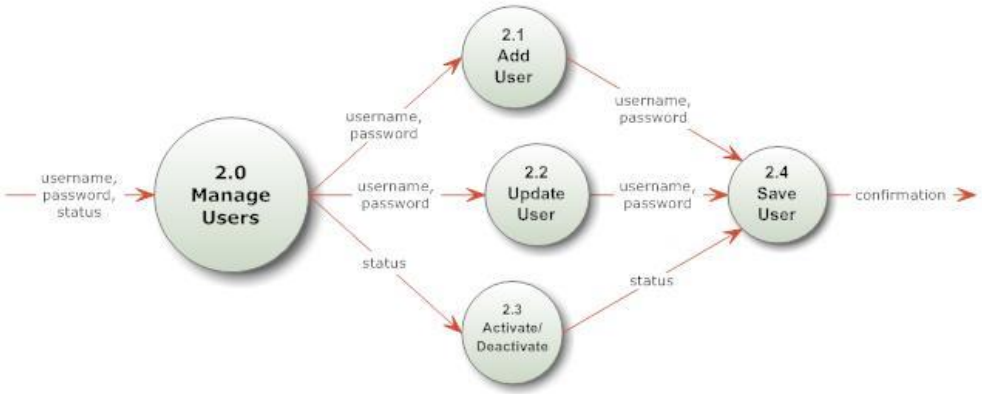

Fig. 10 – Subexplosion for Manage Users

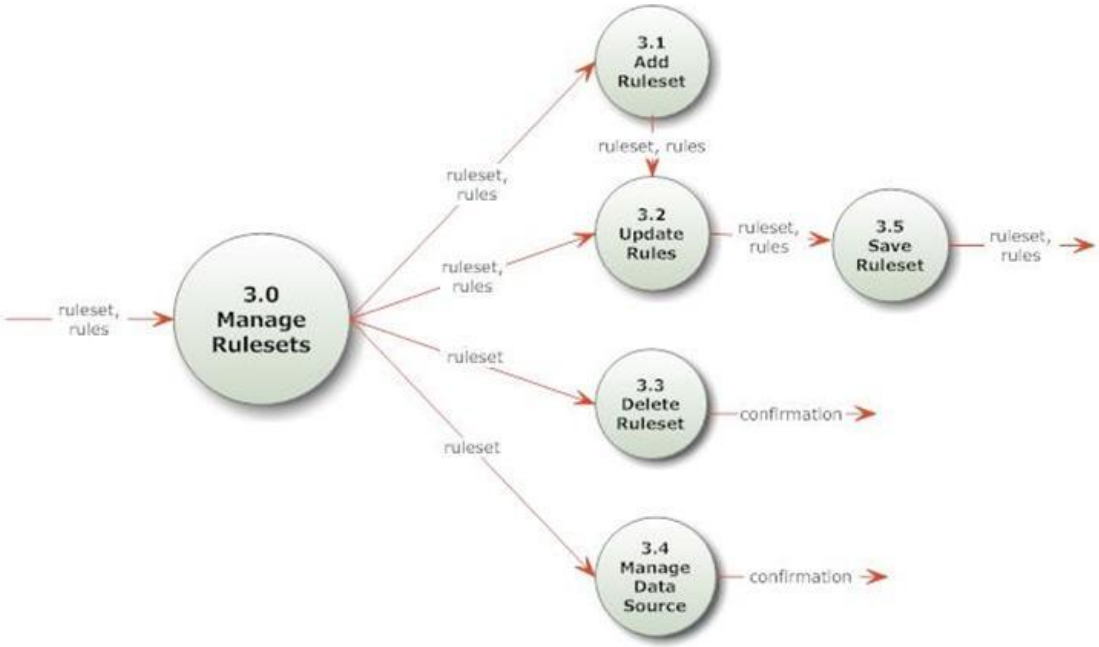

Fig. 11 – Subexplosion for Manage Rulesets

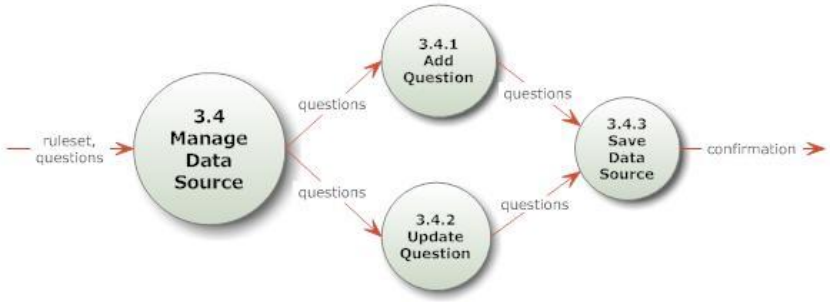

Fig. 12 – Subexplosion for Manage Data Source

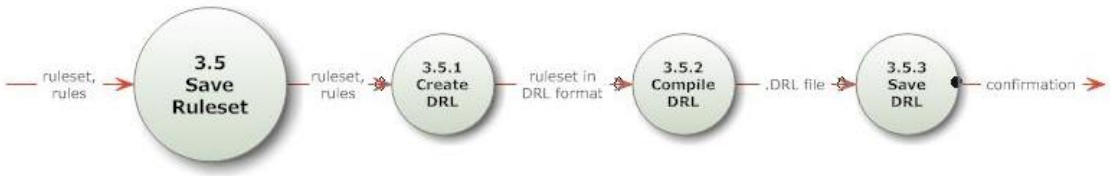

Fig. 13 – Subexplosion for Manage Ruleset – Save Ruleset

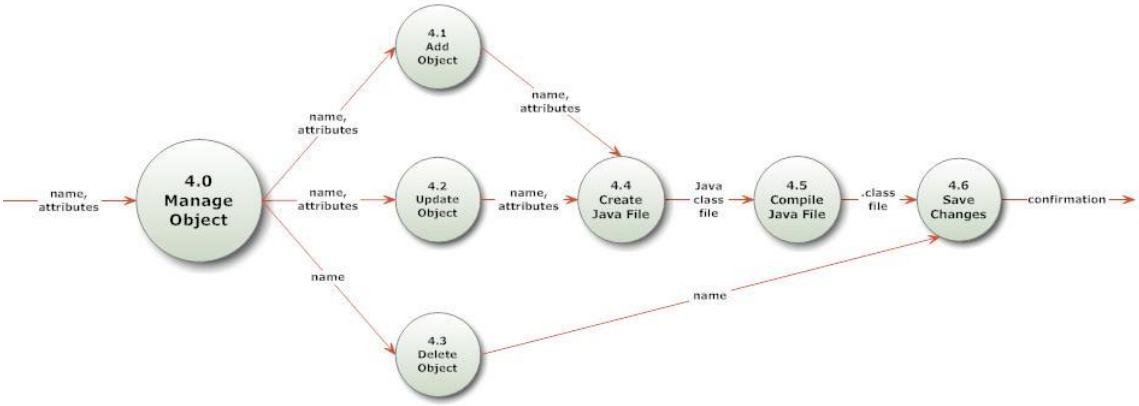

Fig. 14 – Subexplosion for Manage Object

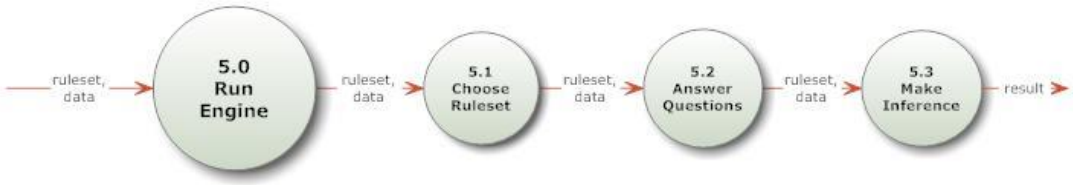

Fig. 15 - Subexplosion for Run Engine

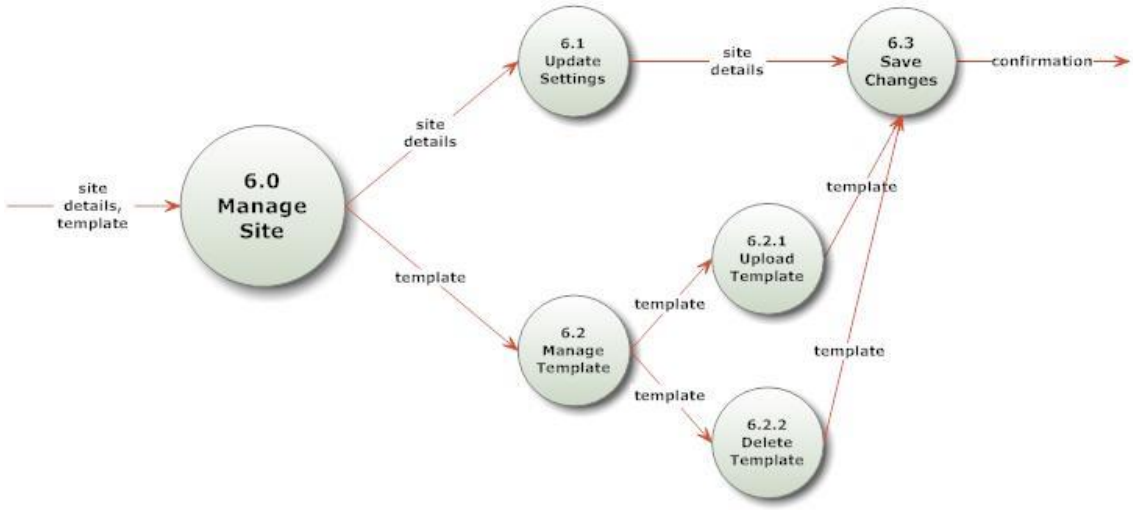

Fig. 16 - Subexplosion for Manage Site

E. Technical Architecture

The system is developed using JSP and MySQL. The system is built using the JSF framework. Other libraries and bin used are: Drools, facelets and RichFaces. The system is deployed on Tomcat localhost for testing purposes.

The following are the list of system requirements:

- Windows (98, XP, Vista) / Linux Operating System
- 256 MB RAM
- 2 MB Disk Space
- Sun JDK 1.5.0_15 or higher
- Application server or web server supporting JSP v1.2 or later (TOMCAT)
- Web browser supporting AJAX (Firefox 3, IE 6, Safari 3)

V. Results and Discussions

GEEBBO provides different functionalities for the owners and target users of the system. The owners, or the business organizations, use the Back-End part of the system for managing the entire system, more specifically, its contents and configurations. The target users, which can either be the clients or customers of the business organization, or the organization itself, use the Front-End part of the system for querying rulesets.

A. Back-End

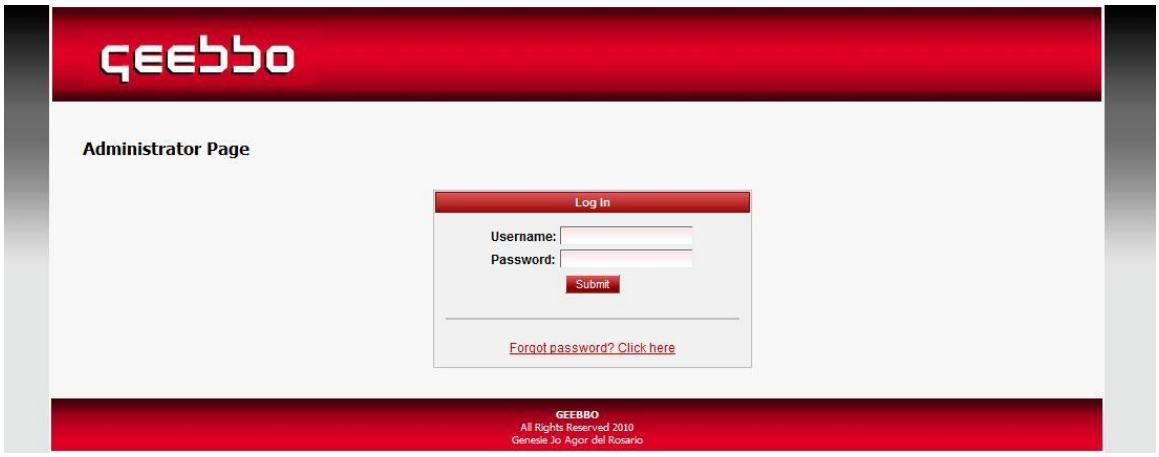

Fig. 17 – Log in

Users need to log in first before using the back-end part of the system. The user is required to give a valid username and password.

Fig. 18 – Password Forgotten

When the user has forgotten his/her password, he/she will be asked a security question he provided (assuming the user has already provided one). When the answer is correct, the password will be set to be the same as the username. It is the responsibility of the user to immediately update his/her password upon logging in the system.

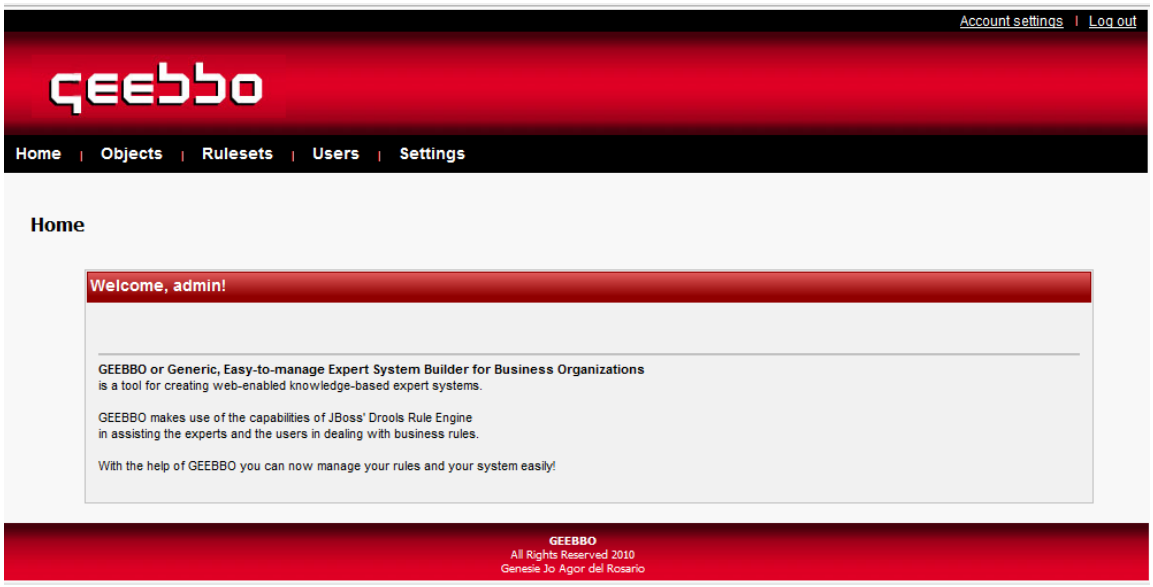

Fig. 19 – Homepage

The homepage contains details about GEEBBO. The menus that are shown are based on user type. The system administrator has the power to view and use all functionalities. The author can only access pages related to rulesets and the programmer can only access pages related to objects and project settings.

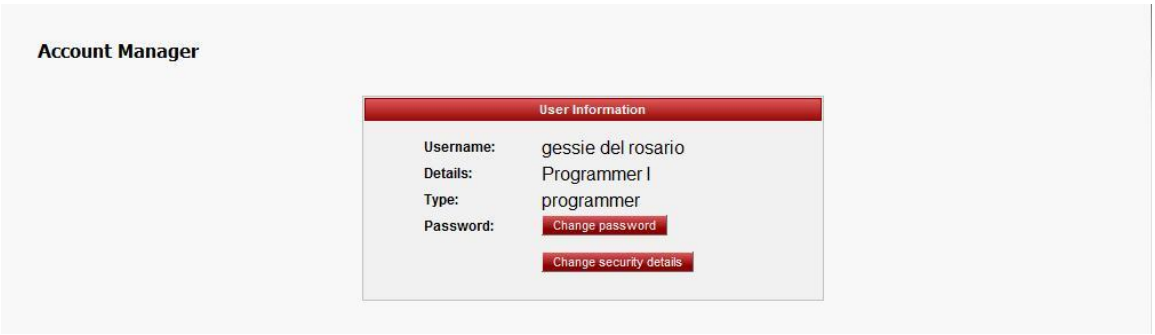

Fig. 20 – Account Settings

Users can view their basic information but they are not allowed to make any changes. This is to ensure validity and authenticity of the users accounts. However, users can still change their passwords (see Fig. 20).

Fig. 21 – Change Password

Fig. 22 – Security Details

A security question and a corresponding answer should be provided by the user. This will be used in case the user has forgotten his/her password.

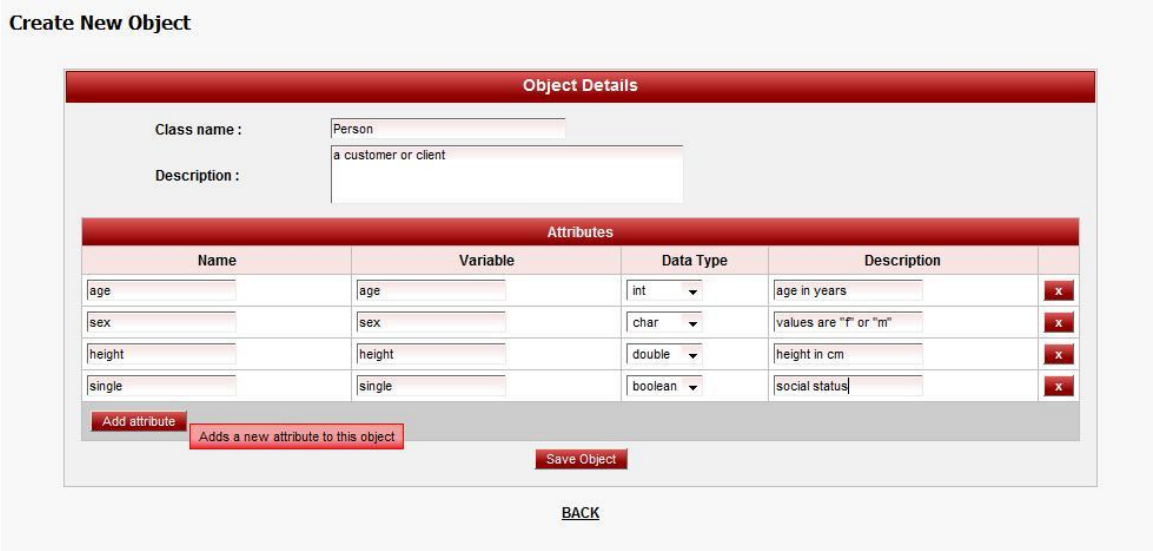

Fig. 23 – Create New Object

Knowledge engineers or programmers are the ones who manage the objects. When objects are created, the class names and attribute variables are validated, i.e., they are ensured not to have special characters and to begin with letters only. When specifying an attribute name, a suggested variable (name with whitespace replaced by underscores) are automatically placed on the textbox, and the user may choose to change this. Attributes are added by clicking the "Add attribute" button. When the "x" button is clicked, that particular attribute is discarded. The object must have at least one attribute.

When the "Save" button is clicked, the data provided by the user are processed to create a java file which will eventually be compiled to a create class file that will be used by the rule engine.

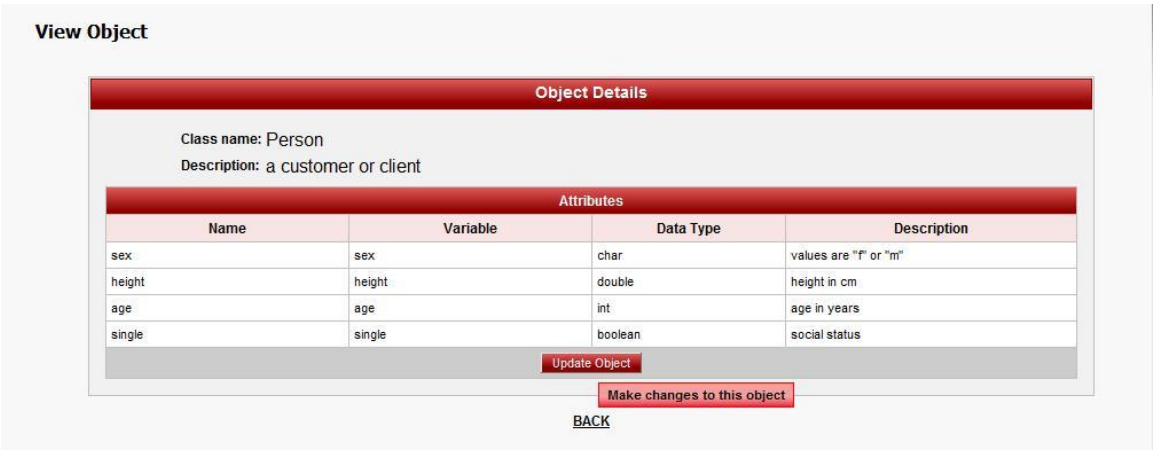

Fig. 24 – Object – View Details

The object details and its attributes are fetched from the database and shown on a non-editable table. When the “Update” button is clicked, the user is redirected to a page similar to the “Create New Object” page, only with the data already provided. The user may add, edit and remove attributes in the same way.

When the “Reset” button is clicked, any changes created by the user is discarded and the original values of the object are restored. When “Save changes” button is clicked, a new java file is created and compiled, overwriting the previous.

Fig. 25 – Object – View

The list of the objects available is presented on a table. The checkboxes on the left specify which objects to delete when the delete button is clicked. When the name of the object is clicked, the user will be redirected to the page for viewing the object details. When the table headers are clicked, the contents of the table will be based on the values under that column.

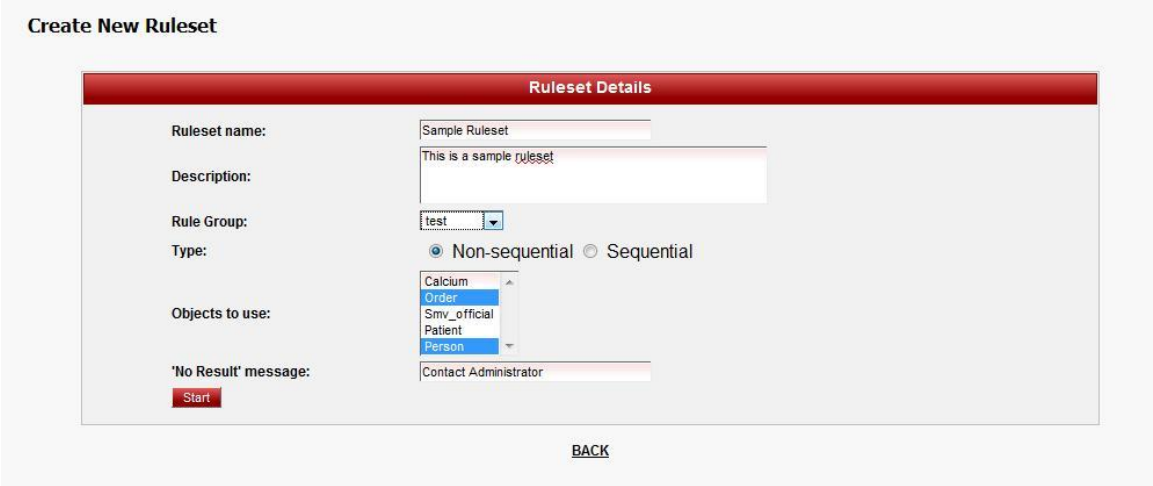

Fig. 26.a – Create New Ruleset

Authors are the ones in charge of feeding rules to the system. The type of ruleset (sequential or nonsequential) and the list of objects to be used are set first and these cannot be changed anymore. The ruleset name is a required field. The author may specify a default message in case the rule engine cannot provide an output.

When the author clicks on start, he/she may start encoding rules and write necessary functions for the ruleset (see fig. 26, b and c).

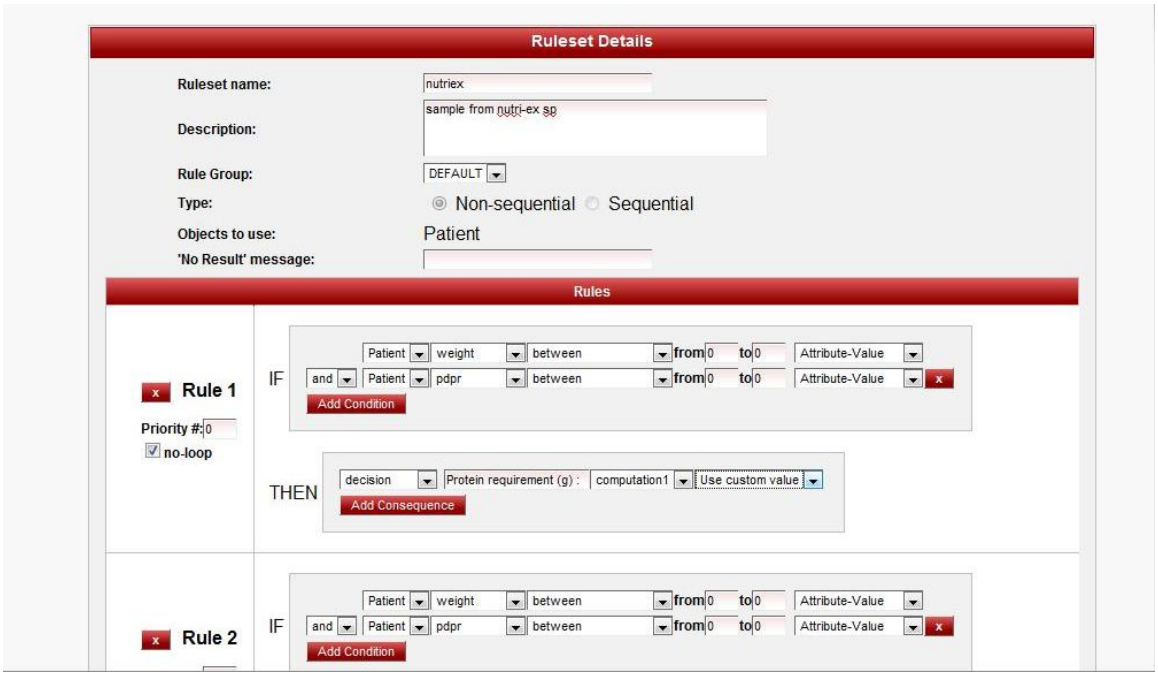

Fig. 26.b– Write Rules

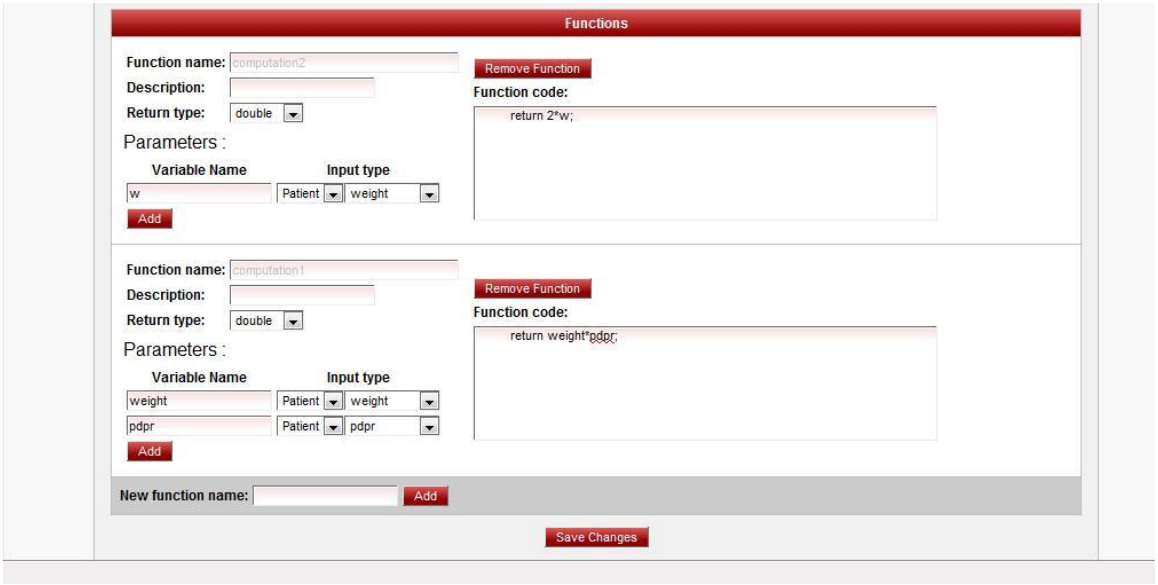

Fig. 26.c– Write Functions

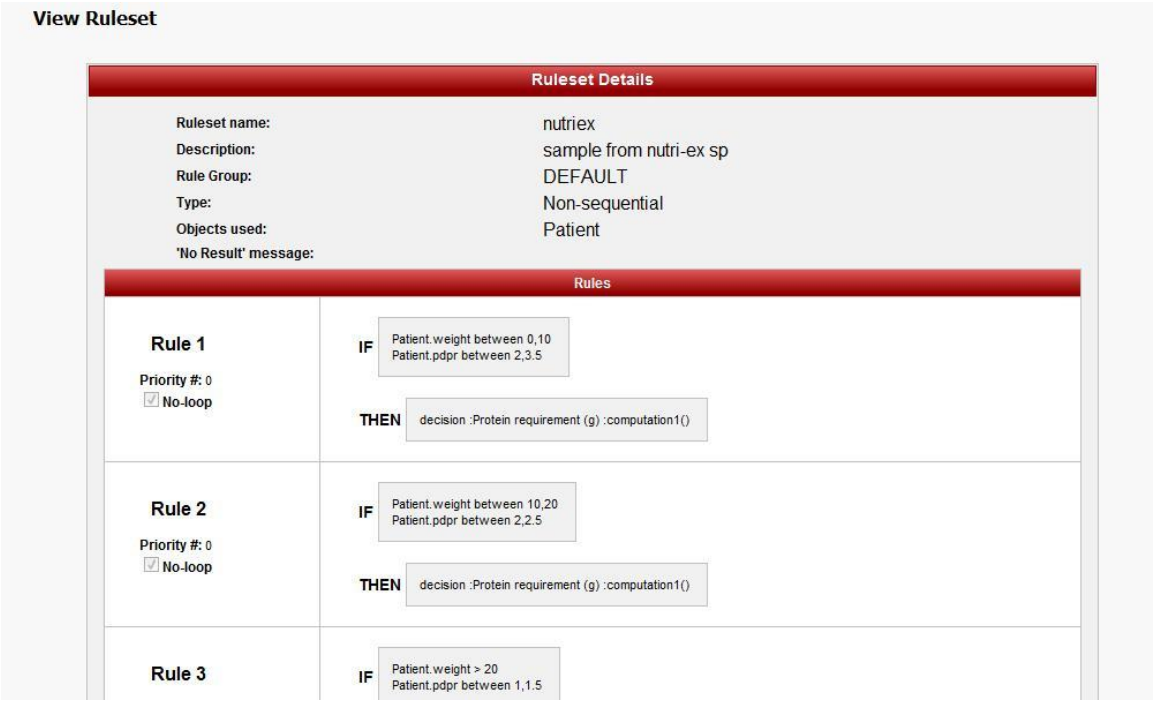

Fig. 27 – Ruleset – View Details

The ruleset details are fetched from the database and shown on a non-editable table. When the “Update” button is clicked, the user is redirected to a page similar to the “Create New Ruleset” page, only with the data already provided. The user may add, edit and remove rules, functions and rule conditions, consequence and attributes.

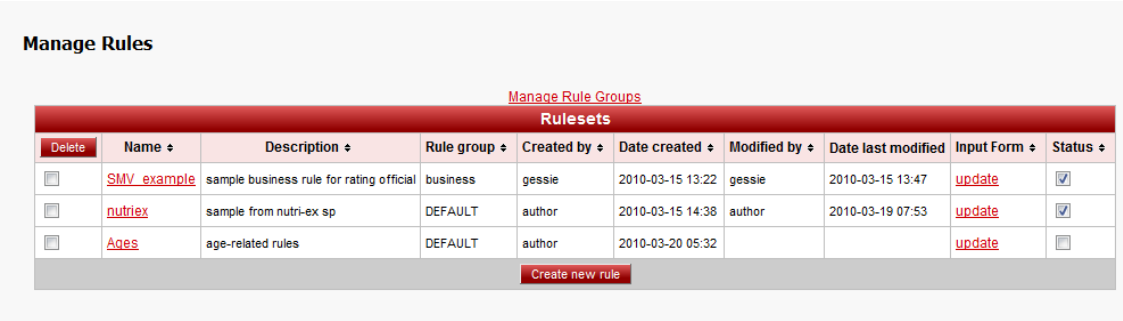

Fig. 28 – Ruleset – View

The list of the rulesets is presented on the table. The checkboxes on the left specify which rulesets to delete when the delete button is clicked. When the name of the object is clicked, the user will be redirected to the page for viewing the Object details. When the checkboxes on the right are clicked, that ruleset will be published (made available to end-users) or unpublished (made unavailable).

When a ruleset is published, a static rule engine instance will be created specifically for that ruleset.

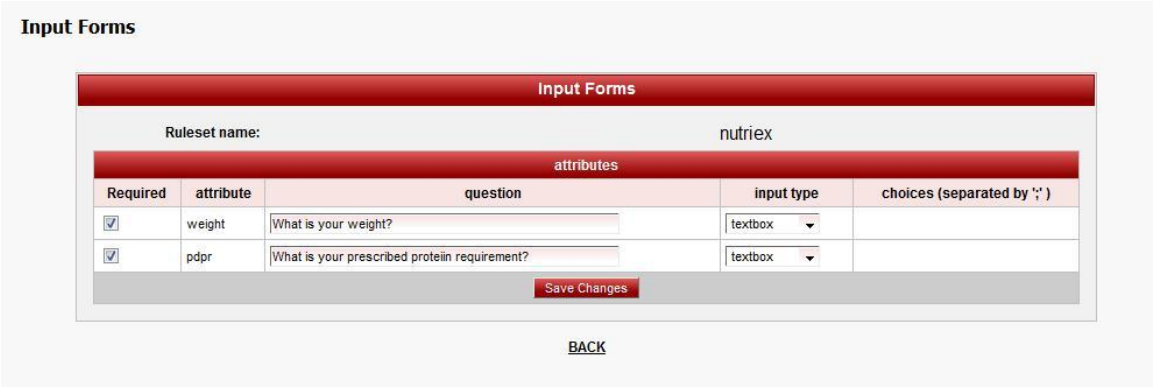

Fig. 29 – Ruleset - Input form

Authors specify how the data will be obtained from the user. The author supplies the question to ask and the type of forms (i.e. textbox, checkbox, dropdown, radio button) to use for each attribute used in the ruleset.

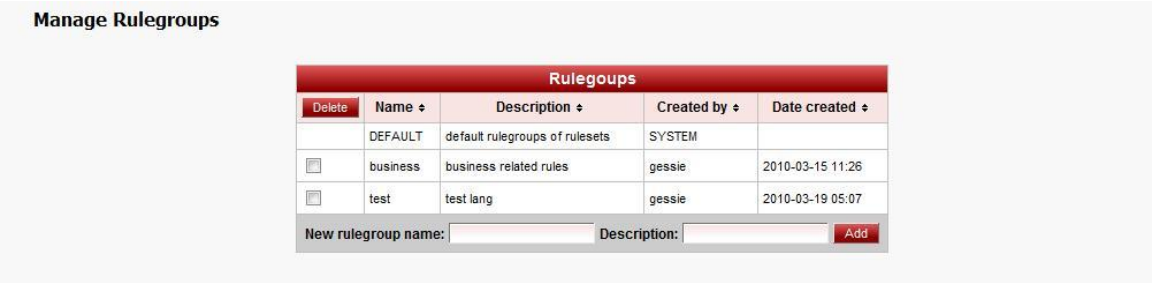

Fig. 30 – Rulegroup

The list of rulegroups is shown on a table. The rulegroups are used for classifying or grouping rulesets according to their types. A new rulegroup is created by providing a name and a description on the bottom of the table and clicking the save button. The checkbox on the left are used for deleting rulegroups (and corresponding rulesets).

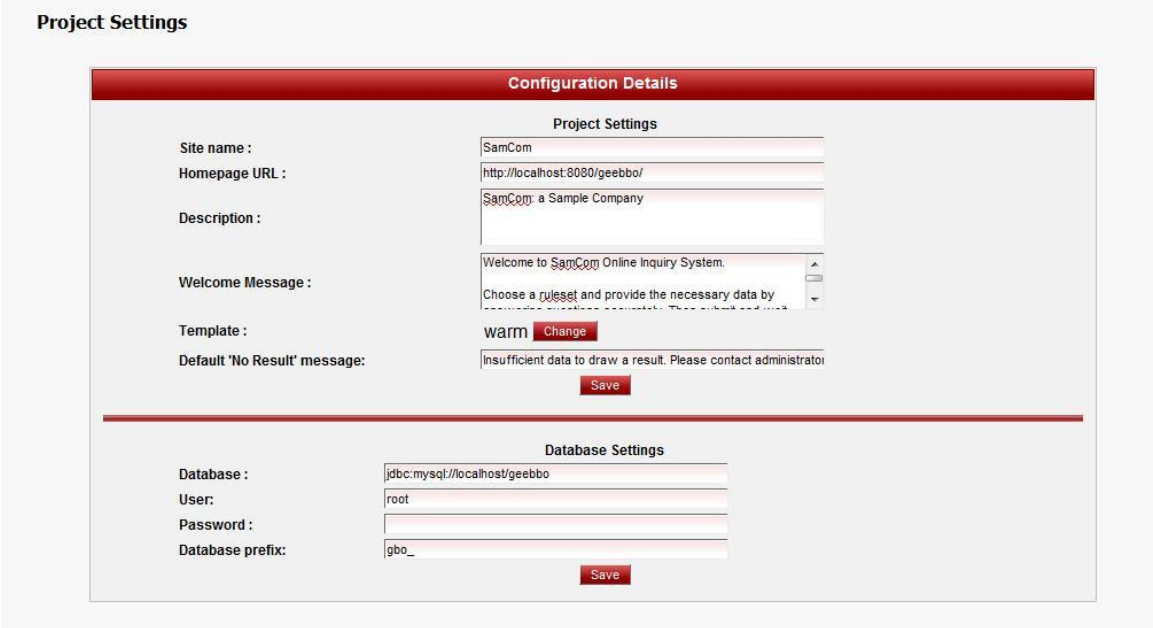

Fig. 31 – ProjectSettings

The programmer and the administrator have the ability to change some of the parameters used by the system, such as basic website details (name of the site, description, welcome message), the template/css to use, and the default message when the Expert system did not come up with a result. Other database-related data, such as the database url, username and password, may also be updated. But note that changes to these parameters may cause the system to malfunction.

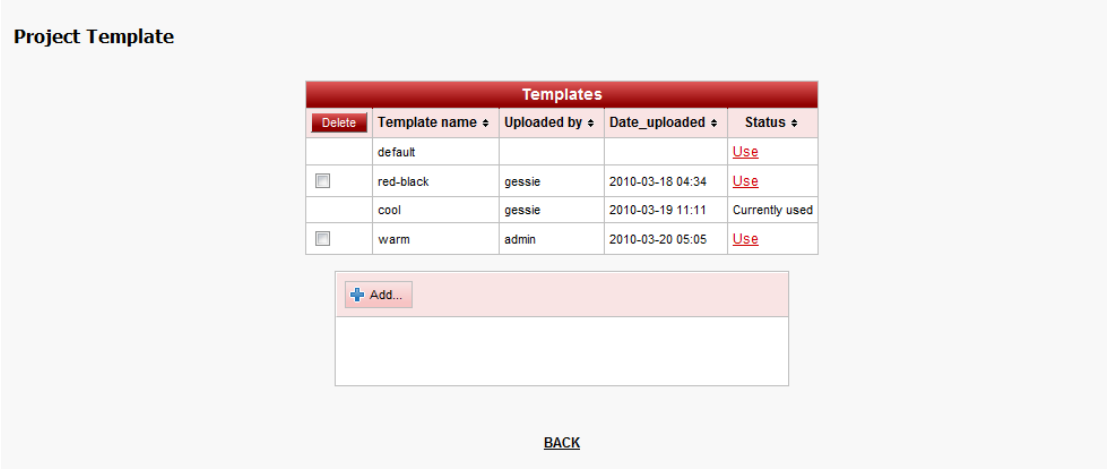

Fig. 32 – Project Template

The programmer and the administrator can upload and set the templates or css files that will be used for the front-end part of the system. They can also delete uploaded templates.

Fig. 33 – View Users

The system administrator can view the list of all the user accounts in the system. He/She can delete users, and activate or deactivate user accounts. He/She can also create and update user accounts.

Fig. 34 – Create New User

The system administrator is the one who provides the user information to make sure that the data provided are authentic.

Fig. 35 – Update User

The administrator can make changes to the user's details. He/She can also reset the password of the users. Resetting a password makes its value same as the username.

B. Front-End

For demonstration purposes, a pseudo-company named "SamCom" (short for SAMple COMpany) is assumed to be using the system and the following are just sample data. Note also

that the appearance of the front-end may vary as it may be changed according to the requirements of the client.

Fig. 36 – Homepage

The front-end part of the system is open to any kind of user. The Homepage simply contains a welcome message and some description about the company and/or website.

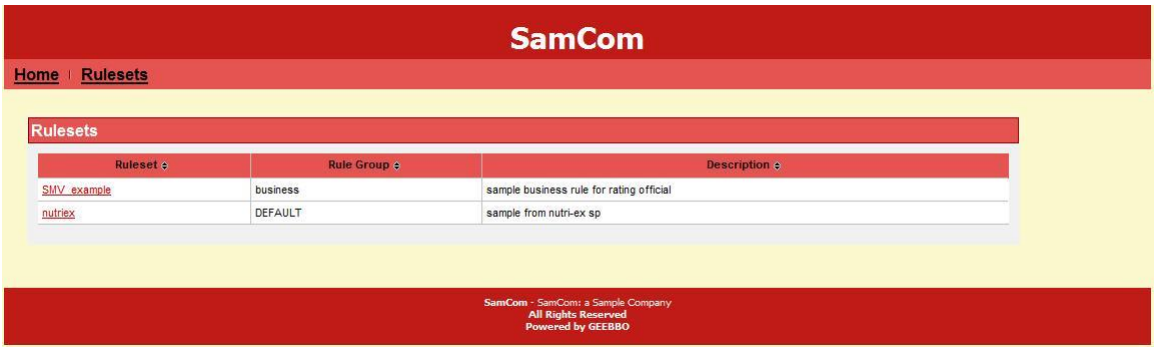

Fig. 37 – View Rulesets

The end-users can view a list of rulesets provided by the business organization or the owner of the system. When the user clicks on the ruleset name, the page is redirected to the page for inputting data.

Fig. 38 – Input Form

The users provide the data by answering the questions. After doing so, the user must click on submit to view the results.

The data submitted by the user are processed and map to the appropriate objects and attributes. These objects are dynamically created and assigned values. These objects will then be given to the static rule engine instances provided by the system.

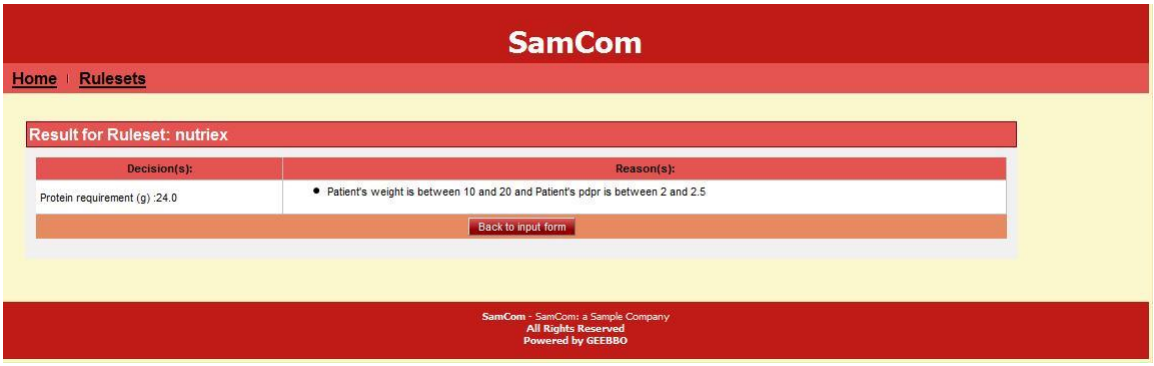

Fig. 39 – View Result

The results are shown in table form, containing the list of decisions that were made while running the ruleset. Together with them are the list of the conditions of the rules that fired which will serve as the explanation or the basis as to how such decision(s) is/are made. Moreover, if the rule outputs a value, the values are also shown on the table.

When the rule engine was not able to make a decision or conclusion, the user is informed through a pre-defined message set by the system administrator or programmer at the back-end.

VI. Conclusion

The Generic, Easy-to-manage Expert System Builder for Business Organizations (or GEEBBO) is able to extend Drools in creating a web-enabled knowledge-based expert system for business organizations. The system is flexible enough and has an easy-to-manage interface which allows the user to customize the system according to their needs.

The System Administrator can log in and log out of the system. The System Administrator can create users and update their information, and can activate and/or deactivate user accounts.

The Author can log in and log out of the system. The Author can create new rulesets, update existing rulesets and delete rulesets. The author manages the data source for each rulesets by specifying what data are needed and how to ask for data.

The Knowledge Engineer or Programmer can log in and log out of the system. The Knowledge Engineer can create new objects, update existing objects and delete objects.

The End-user of the system can view the list of published ruleset. The End-user can choose which ruleset to query, provide the data or facts needed by the expert system and then view the result of the query.

Moreover, customization of the look and feel of the system based on the needs and specifications of the client (the business organizations) is made easy through changing parameter details and uploading templates.

VII. Recommendations

The following are some suggestions and recommendations that can be done to further improve the system:

The system can be made embeddable to other websites. Instead of being an independent web application, the client may choose to simply make the system a part of an existing company website.

More validations can also be implemented when creating objects. The object details must be assured to strictly follow Java conventions.

The page for updating rules can be made more user-friendly by adding a “Help” function which the user may check when he/she has questions or troubles with the meanings and functionalities of each element in the page.

VIII. Bibliography

[1] "expert system." Encyclopædia Britannica. 2009. Encyclopædia Britannica Online. 12 Oct. 2009 <http://www.britannica.com/EBchecked/topic/198506/expert-system>. Grace, R. “Expert Systems for Lithic Analysis.” Online. 12 Oct. 2009 <http://www.hf.uio.no/iakh/forskning/sarc/iakh/lithic/expsys.html#anchor130219>

[2] Vincent P. “Business Rules Management and BPM, Who’s Managing Your Rules?” 2005. Fair Isaac. Online. 12 Oct. 2009. <http://www.theorsociety.com/region/study/infor/BUSINESSRULESMGMTANDBPM.pdf>

[3] Graham, I. “Aligning IT with Business.” Business Rules Management and Service Oriented Architecture: A Pattern Language. 2007.

[4] “Why would I use a Business Rules Management System?” RealRules Blogzine. 2006. RealRules Blogzine. 9 Oct. 2009 < http://oxygen.informatik.tu-cottbus.de/RealRules/?q=node/8>.

[5] “FICO™ Blaze Advisor® business rules management.” 2007. FICO™. Online. 12 Oct. 2009. <http://www.fico.com/en/Products/DMTools/Pages/FICO-Blaze-Advisor-System.aspx>

[6] “ILOG Business Rule Management Systems”. IBM ILOG Products and Solutions. 2009. IBM. Online 12 Oct 2009. <http://www.ilog.com/products/businessrules/>

[7] Grahan, Ian. “Service Oriented Business Rules Management Systems.” 2006. TriTeme International Ltd. 9 Oct 2009 <http://www.trireme.com/Service_Oriented_Business_Rules_Management_Systems.htm>

[8] Owen, J. “Open source rule management.” 2006. JavaWorld. Online. 26 Oct 2009. <http://www.javaworld.com/javaworld/jw-11-2006/jw-1107-brms.html>

[9] “Open Source Business Rules Management”. 2007. OpenRules. Online. 12 Oct 2009. <http://openrules.com/index.htm>

[10] “Drools Guvnor.” JBoss Community. 2009. JBoss Community. Online. 2 Oct 2009. . <http://www.jboss.org/drools/drools-guvnor.html>

[11] “Agility”. Savvion. 2009. Savvion. Online. 9 Oct 2009. <http://www.savvion.com/agility>

[12] Holloway, S. “Bloor’s first report on Business Rules Management Systems.” Bloor. 2009. Bloor. Onlilne. 9 Oct. 2009. <http://www.it-director.com/business/content.php?cid=11547>

[13] Taylor, J. “The difference between business rules and expert systems”. informIT. 2007. informIT. Online. 9 Oct 2009. <http://www.informit.com/blogs/blog.aspx?uk=The-difference-between-business-rules-and-expert-systems>

[14] Kumar, B.R., Shanmugam, J., Janarthanan, S. and Santhiseela, R. “A Generic component based expert system shell for airborne equipment design.” Madras Institute of Technology, Chennai, Tamil Nadu, India. 2004.

[15] Sartikai, Y., Suwardi, I. S. “Generic Expert System – Incremental Learning System”. Proceedings of the International Conference on Electrical Engineering and Informatics. Bandung Institute of Technology. 2007.

[16] Harrington, R.A., Banks, S., and Santos, E. Jr. “GESIA: Uncertainty-Based Reasoning for a Generic Expert System Intelligent User Interface”. Proceedings of the 8th IEEE International Conference on Tools with Artificial Intelligence. 1996.

[17] Lovrencic, S., Rabuzin, K., Picek, R. “Formal Modelling of Business Rules: What Kind of Tool to Use?” Journal of Information and Organizational Sciences. 2006. Vol.30. Issue 2, pp. 23-27.

[18] Andreescu, A. L., Uta, A. “Methodological approaches based on business rules.” Informatica Economica Journal. 2008. Vol.XII. Issue 3, pp. 23-27.

[19] Avram, V., Avram, D. “Managing Knowledge within the Small and Midsized Companies.” Informatica Economica Journal. 2007. Vol.XI. Issue 4, pp. 33-36.

[20] Smeureanu, I., Diosteanu, A. “A Collaborative System Software Solution for Modeling Business Flows Based on Automated Semantic Web Service Composition”. Informatica Economica Journal. 2009. Vol.13. Issue 2, pp. 32-40.

[21] Brad, S. “Mobil Digital Management for SME’s.” Theoretical and Applied Economics. 2008. Vol. 10(527). Issue 10(527), pp87-92.

[22] Aqel, M. J. “An Approach for Developing Uncertain Multi-perspective Expert System Using the Algebraic Sum Method.” American Journal of Applied Sciences. 2006. 3(2): 1719-1721.

[23] De Sordi, J. O., Spelta, A. G. “Business process management systems technology components analysis.” Journal of Information Systems and Technology Management. 2006. Vol.4, No. 1.

[24] “Business Rules”. Agile Modeling. 2009. Agile Modeling. 9 Oct. 2009. <http://www.agilemodeling.com/essays/businessRule.htm>

[25] “Business rules engine.” Wikipedia. 2009. Wikipedia. 12 Oct 2009. <http://en.wikipedia.org/wiki/Business_rules_engine>.

[26] Mahmoud, Gusay H. “Getting Started With the Java Rule Engine API (JSR 94): Toward Rule-Based Applications”. Sun Developer Network. 2009. Sun Developer Network. 2 Oct 2009. <http://java.sun.com/reference/techart/JavaRule.html>

[27] “Drools 5 – The Business Logic integration Platform.” JBoss Community. 2009. JBoss Community. 2 Oct. 2009. <http://www.jboss.org/drools/>

[28] “Drools Introduction and General User Guide.” JBoss Community. 2009. JBoss Community. 2 Oct 2009. <http://www.jboss.org/drools/documentation.html>

[29] “Drools Expert User Guide.” JBoss Community. 2009. JBoss Community. 2 Oct 2009. .
<http://www.jboss.org/drools/drools-expert.html>.

IX. Appendix

/.classpath

```
<?xml version="1.0" encoding="UTF-8"?>
<classpath>
   <classpathentry kind="src" path="WEB-INF/classes"/>
   <classpathentry kind="con"
path="org.eclipse.jdt.launching.JRE_CONTAINER"/>
   <classpathentry kind="var" path="JRE_LIB" sourcepath="JRE_SRC"/>
   <classpathentry kind="lib" path="WEB-INF/lib/mysql-connector-java-5.1.10-
bin.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/antlr-runtime-3.1.1.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/commons-beanutils-
1.8.0.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/commons-collections-
3.2.1.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/commons-digester-2.0.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/commons-el.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/commons-fileupload.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/commons-io-1.1.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/commons-logging-1.1.1.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/core-3.4.2.v_883_R34x.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/drools-api-5.0.1.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/drools-compiler-5.0.1.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/drools-core-5.0.1.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/drools-jsr94-5.0.1.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/el-ri.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/jhighlight-1.0.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/jsf-api.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/jsf-facelets.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/jsf-impl.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/mvel2-2.0.10.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/richfaces-api-3.3.2.SR1.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/richfaces-impl-3.3.2.SR1.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/richfaces-ui-3.3.2.SR1.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/annotations-api.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/catalina.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/catalina-ant.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/catalina-ha.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/catalina-tribes.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/el-api.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/jasper.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/jasper-el.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/jasper-jdt.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/jsp-api.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/servlet-api.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/tomcat-coyote.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/tomcat-dbcp.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/tomcat-i18n-es.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/tomcat-i18n-fr.jar"/>
   <classpathentry kind="lib" path="WEB-INF/lib/tomcat-i18n-ja.jar"/>
   <classpathentry kind="output" path="WEB-INF/classes"/>
</classpath>
```

/home.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:a4j="http://richfaces.org/a4j"
      xmlns:rich="http://richfaces.org/rich">
<ui:composition template="includes/template.xhtml">
   <ui:define name="title"><h:outputText value="#{page.siteName}"
/></ui:define>
   <ui:define name="content">
  <rich:panel styleClass="panel_content" headerClass="panel_header">

   <f:facet name="header">  <h:outputText value="Home Page" />  </f:facet>
   <h:outputText value="#{page.siteMsg}" />
  </rich:panel>   </ui:define>  </ui:composition></html>
```

/index.jsp

```
<jsp:forward page="home.jsf" />
```

/admin/forgot.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:a4j="http://richfaces.org/a4j"
      xmlns:rich="http://richfaces.org/rich">
<ui:composition template="includes/template.xhtml">
  <ui:define name="title">Administrator Page</ui:define>
  <ui:define name="content">  <f:view>  <h:form>
       <rich:panel styleClass="table_login">
         <f:facet name="header" ><h:outputText value="Password Forgot" />
</f:facet>
        <h:panelGrid columns="2">
           <h:outputText value="Username: " styleClass="text_label"/>
           <h:outputText value="#{user.username}" />
           <h:outputText value="#{user.question}"  styleClass="text_label"/>
           <h:inputText value="#{user.ansVer}" />
        </h:panelGrid>
        <h:panelGrid columns="1">
          <h:commandButton action="#{user.checkForgot}" value="Submit" />
        </h:panelGrid>
       </rich:panel>  </h:form> </f:view> </ui:define> </ui:composition> </html>
```

/admin/home.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:rich="http://richfaces.org/rich">
<ui:composition template="includes/template.xhtml">
  <ui:define name="title">Home</ui:define>
  <ui:define name="content">
  <rich:panel styleClass="home_panel"  headerClass="home_header">
    <f:facet name="header">
      <h:outputText value="Welcome, #{sessionbean.username}!" />
    </f:facet>    <br/><br/>    <hr /><b>
    GEEBBO or Generic, Easy-to-manage Expert System Builder for Business
Organizations </b><br/>    is a tool for creating web-enabled knowledge-based
expert systems. <br/><br/>  GEEBBO makes use of the capabilities of JBoss'
Drools Rule Engine <br/>  in assisting the experts and the users in dealing with
business rules. <br /><br/>  With the help of GEEBBO you can now manage
your rules and your system easily! <br/><br/>
  </rich:panel>  </ui:define> </ui:composition> </html>
```

/admin/index.jsp

```
<jsp:forward page="home.jsf" />
```

/admin/login.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:a4j="http://richfaces.org/a4j"
      xmlns:rich="http://richfaces.org/rich">
<ui:composition template="includes/template.xhtml">
  <ui:define name="title">Administrator Page</ui:define>
  <ui:define name="content">
  <f:view>  <h:form>
       <rich:panel styleClass="table_login">
         <f:facet name="header" > <h:outputText value="Log In" />  </f:facet>
         <h:panelGrid columns="2">
            <h:outputText value="Username: " styleClass="text_label"/>
```

```
          <h:inputText value="#{user.username}" required="true"
requiredMessage="Username must have a value"/>
          <h:outputText value="Password:"  styleClass="text_label"/>
          <h:inputSecret value="#{user.password}" />
        </h:panelGrid>
        <h:panelGrid columns="1">
          <h:commandButton action="#{user.login}" value="Submit" />
        </h:panelGrid>    <br />    <hr />     <br />
        <a4j:commandLink value="Forgot password? Click here "
action="#{user.pwdForgot}">
          <rich:toolTip value="Answer the security question and reset password"
styleClass="tooltip"/>
        </a4j:commandLink> </rich:panel> </h:form>  </f:view>  </ui:define>
  </ui:composition> </html>
```

/admin/restricted.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets">
<ui:composition template="includes/template.xhtml">
  <ui:define name="title">Restricted Access</ui:define>
  <ui:define name="content">
  <center>You are not authorized to view this page.</center><br />
  </ui:define></ui:composition></html>
```

/admin/account/account.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:a4j="http://richfaces.org/a4j"
      xmlns:rich="http://richfaces.org/rich">
<ui:composition template="../includes/template.xhtml">
  <ui:define name="title">Account Manager</ui:define>
  <ui:define name="content">
  <f:view><h:form>
      <rich:panel styleClass="table_account">
        <f:facet name="header" >
          <h:outputText value="User Information" />
        </f:facet>
        <h:panelGrid columns="2" styleClass="panel_details">
          <h:outputText value="Username: "  styleClass="text_label"/>
          <h:outputText value="#{user.name}" />
          <h:outputText value="Details: "  styleClass="text_label"/>
          <h:outputText value="#{user.details}" />
          <h:outputText value="Type: "  styleClass="text_label"/>
          <h:outputText value="#{user.stringType}" />
          <h:outputText value="Password:"  styleClass="text_label"/>
          <a4j:commandButton action="#{page.changePwd}" value="Change
password">
            <rich:toolTip styleClass="tooltip" value="Click to change password"/>
          </a4j:commandButton><br /><br />
          <a4j:commandButton action="#{page.changeQA}" value="Change
security details">
            <rich:toolTip styleClass="tooltip" value="Click to change account
security details"/>
          </a4j:commandButton>
        </h:panelGrid</rich:panel> </h:form> </f:view> </ui:define>
</ui:composition> </html>
```

/admin/account/changePwd.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:a4j="http://richfaces.org/a4j"
      xmlns:rich="http://richfaces.org/rich">
<ui:composition template="../includes/template.xhtml">
  <ui:define name="title">Account Manager</ui:define>
  <ui:define name="content">
  <f:view><h:form>
      <rich:panel styleClass="table_account">
        <f:facet name="header" >
          <h:outputText value="Change Password" />
        </f:facet>
        <h:panelGrid columns="2" styleClass="panel_details">
          <h:outputText value="Current password: "  styleClass="text_label"/>
          <h:inputSecret value="#{user.pwdCur}" />
          <h:outputText value="New password: "  styleClass="text_label"/>
          <h:inputSecret value="#{user.pwdNew}" />
          <h:outputText value="Verify new password : "
styleClass="text_label"/>
          <h:inputSecret value="#{user.pwdVer}" /> </h:panelGrid>
        <h:panelGrid columns="1">
          <a4j:commandButton action="#{user.changePwd}" value="Change
password" >
            <rich:toolTip styleClass="tooltip" value="Save new password"/>
          </a4j:commandButton>
        </h:panelGrid></rich:panel>
  <div id="link_back">    <br/>
    <a4j:commandLink value="BACK" action="#{page.viewAccount}"
styleClass="link_back">
      <rich:toolTip styleClass="tooltip" value="Go back to user information" />
    </a4j:commandLink>
  </div></h:form></f:view></ui:define></ui:composition></html>
```

/admin/account/changeQA.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:a4j="http://richfaces.org/a4j"
      xmlns:rich="http://richfaces.org/rich">
<ui:composition template="../includes/template.xhtml">
  <ui:define name="title">Account Manager</ui:define>
  <ui:define name="content">
  <f:view><h:form>
      <rich:panel styleClass="table_account">
        <f:facet name="header" >
          <h:outputText value="Change Security Details" />
        </f:facet>
        <h:outputText value="Provide a question and answer to be used when
password is forgotten." /> <br />
        <h:panelGrid columns="2" styleClass="panel_details">
          <h:outputText value="Security question: "  styleClass="text_label"/>
          <h:inputTextarea value="#{user.question}" />
          <h:outputText value="Answer: "  styleClass="text_label"/>
          <h:inputTextarea value="#{user.answer}" /></h:panelGrid>
        <h:panelGrid columns="1">
          <a4j:commandButton action="#{user.changeQA}" value="Update
details" >
            <rich:toolTip styleClass="tooltip" value="Save details"/>
          </a4j:commandButton></h:panelGrid> </rich:panel>
  <div id="link_back">  <br/>
    <a4j:commandLink value="BACK" action="#{page.viewAccount}"
styleClass="link_back">
      <rich:toolTip styleClass="tooltip" value="Go back to user information" />
    </a4j:commandLink></div></h:form></f:view>
</ui:define></ui:composition></html>
```

/admin/includes/footer.xhtml

```
<ui:composition xmlns:f="http://java.sun.com/jsf/core"
                xmlns:h="http://java.sun.com/jsf/html"
                xmlns:ui="http://java.sun.com/jsf/facelets">
<f:view><b>GEEBBO</b><br />All Rights Reserved 2010<br />Genesie Jo
Agor del Rosario</f:view></ui:composition>
```

/admin/includes/header.xhtml

```
<ui:composition xmlns:ui="http://java.sun.com/jsf/facelets">
<div id="header">GEEBBO</div></ui:composition>
```

/admin/includes/menu.xhtml

```
<ui:composition xmlns:f="http://java.sun.com/jsf/core"
                xmlns:h="http://java.sun.com/jsf/html"
```

```
            xmlns:ui="http://java.sun.com/jsf/facelets"
            xmlns:rich="http://richfaces.org/rich"
            xmlns:a4j="http://richfaces.org/a4j">
<f:view><h:form>
<rich:toolBar styleClass="menu_sub" rendered="#{sessionbean.loggedIn}">
<rich:toolBarGroup itemSeparator="line" location="right">
  <a4j:commandLink action="#{page.viewAccount}" value="Account settings"
rendered="#{sessionbean.loggedIn}" styleClass="menu_sub">
    <rich:toolTip value="View account details" styleClass="tooltip"/>
  </a4j:commandLink>
  <a4j:commandLink action="#{page.logout}" value="Log out"
rendered="#{sessionbean.loggedIn}" styleClass="menu_sub">
    <rich:toolTip value="Log out of the system" styleClass="tooltip"/>
  </a4j:commandLink>
</rich:toolBarGroup>
</rich:toolBar>
<div class="header">
  <h:graphicImage value="/admin/templates/default/geebbo2.jpg"
alt="GEEBBO"/></div>
<rich:toolBar itemSeparator="line" styleClass="menu_main"
rendered="#{sessionbean.loggedIn}">
  <rich:dropDownMenu value="Home" action="#{page.viewHome}"
styleClass="menu_dropdown"  >
    <rich:menuItem value="Home" action="#{page.viewHome}"
styleClass="menu_item"/>
  </rich:dropDownMenu>
  <rich:dropDownMenu value="Objects" styleClass="menu_dropdown"
rendered="#{sessionbean.type!=1}">
    <rich:menuItem value="View"  action="#{page.viewObjects}"
styleClass="menu_item"/>
    <rich:menuItem value="Create new"  action="#{page.addObject}"
styleClass="menu_item"/>
  </rich:dropDownMenu>
  <rich:dropDownMenu value="Rulesets" styleClass="menu_dropdown"
rendered="#{sessionbean.type!=2}">
    <rich:menuItem value="View"  action="#{page.viewRules}"
styleClass="menu_item"/>
    <rich:menuItem value="Create new"  action="#{page.addRuleset}"
styleClass="menu_item"/>
    <rich:menuItem value="Manage rulegroups"
action="#{page.viewRulegroups}" styleClass="menu_item"/>
  </rich:dropDownMenu>
  <rich:dropDownMenu value="Users" styleClass="menu_dropdown"
rendered="#{sessionbean.type==3}">
    <rich:menuItem value="View"  action="#{page.viewUsers}"
styleClass="menu_item"/>
    <rich:menuItem value="Create new"  action="#{page.addUser}"
styleClass="menu_item"/>
  </rich:dropDownMenu>
  <rich:dropDownMenu value="Settings" styleClass="menu_dropdown"
rendered="#{sessionbean.type==3}">
    <rich:menuItem value="View" action="#{page.viewSettings}"
styleClass="menu_item"/>
    <rich:menuItem value="Manage templates"
action="#{page.viewTemplates}" styleClass="menu_item"/>
  </rich:dropDownMenu></rich:toolBar></h:form></f:view></ui:composition>
```

/admin/includes/template.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  xmlns:f="http://java.sun.com/jsf/core"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:c="http://java.sun.com/jsp/jstl/core"
  xmlns:a4j="http://richfaces.org/a4j"
  xmlns:rich="http://richfaces.org/rich">
<head><title>GEEBBO - <ui:insert name="title" /></title>
    <link rel="stylesheet" type="text/css"
href="/geebbo/admin/templates/#{page.backTemplate}" /></head>
<body><center>
<table class="main_body">
  <tr><td> <ui:include src="menu.xhtml"/> </td></tr>
  <tr><td class="content">
    <p class="title"> <ui:insert name="title" /> </p>
    <center>
    <rich:messages errorClass="errorMessage" infoClass="infoMessage"
     layout="table" showDetail="false"
     showSummary="true" id="hMessage"/><br />
       <ui:insert name="content" />
    </center> <br/>  </td></tr>
  <tr><td class="footer"><ui:include src="footer.xhtml"/></td></tr>
</table></center></body></html>
```

/admin/object/add.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:a4j="http://richfaces.org/a4j"
    xmlns:rich="http://richfaces.org/rich">
<ui:composition template="../includes/template.xhtml">
    <ui:define name="title"><h:outputText value="#{object.title}" /></ui:define>
    <ui:define name="content"><f:view><h:form>
            <rich:panel headerClass="view_header" styleClass="panel_main"
id="panel_main">
                <f:facet name="header">
                    <h:outputText value="Object Details" /></f:facet>
                <h:panelGrid columns="2" styleClass="panel_details" >
                    <h:outputText value="Class name : " styleClass="text_label" />
                <h:inputText value="#{object.name}" styleClass="textbox_name"/>
                    <h:outputText value="Description :" styleClass="text_label" />
                    <h:inputTextarea value="#{object.description}"
                        styleClass="textarea_desc" /></h:panelGrid>
                <h:panelGrid styleClass="panel_contents">
<!-- INPUT ATTRIBUTES -->
                    <rich:dataTable value="#{object.attrs}" var="attr"
                        id="datatable" styleClass="panel_datatable"
            headerClass="view_subheader" footerClass="view_footer">
                        <f:facet name="header">
                            <h:outputText value="Attributes" />    </f:facet>
                        <rich:column>
                            <f:facet name="header">
                                <h:outputText value="Name" /></f:facet>
                            <h:inputText value="#{attr.name}"
rendered="#{!attr.removed}">
                                <a4j:support event="onchange"
action="#{attr.updateVariable}" reRender="datatable" /></h:inputText>
                        </rich:column>
                <rich:column>
                  <f:facet name="header">
                    <h:outputText value="Variable" /></f:facet>
                  <h:inputText value="#{attr.variable}" rendered="#{!attr.removed}" />
                </rich:column>
                        <rich:column>
                            <f:facet name="header">
                                <h:outputText value="Data Type" /></f:facet>
                            <h:selectOneMenu value="#{attr.datatype}"
                                rendered="#{!attr.removed}">
                                <f:selectItems value="#{object.selectDataTypes}" />
                            </h:selectOneMenu></rich:column>
                        <rich:column>
                            <f:facet name="header">
                                <h:outputText value="Description" />
                            </f:facet>
                            <h:inputText value="#{attr.description}"
                                rendered="#{!attr.removed}" /></rich:column>
                        <rich:column styleClass="delete_column">
                            <a4j:commandButton value="x" reRender="datatable"
                                styleClass="button_delete"
rendered="#{!attr.removed}"action="#{attr.removeThis}"   onclick="if
(!confirm('#{object.askDelAtt}')) return false">
```

```
                    <rich:toolTip value="Remove attribute"
styleClass="tooltip"/></a4j:commandButton></rich:column>
                  <f:facet name="footer">
                    <h:panelGrid columns="2">
                      <a4j:commandButton value="Add attribute"
reRender="datatable" action="#{object.addAttribute}">
                        <rich:toolTip value="Adds a new attribute to this
object" styleClass="tooltip"/></a4j:commandButton>
                      <a4j:commandButton action="#{object.init}" value="Reset
values" rendered="#{object.editAttr}" reRender="panel_main">
                    <rich:toolTip value="Discards changes made and brings back
original values" styleClass="tooltip" /></a4j:commandButton></h:panelGrid>
</f:facet> </rich:dataTable></h:panelGrid>
                <h:panelGrid styleClass="panel_footer">
                  <a4j:commandButton action="#{object.save}" value="Save
Object" rendered="#{!object.editAttr}">
                    <rich:toolTip value="Save and creates this object"
styleClass="tooltip"/></a4j:commandButton>
          <a4j:commandButton action="#{object.update}" value="Save Changes"
rendered="#{object.editAttr}"  onclick="if (!confirm('Affected rulesets will be
automatically deleted. Are you sure you want to continue?')) return false">
            <rich:toolTip value="Updates and recreates this object"
styleClass="tooltip"/></a4j:commandButton></h:panelGrid></rich:panel>
            <div id="link_back"><br />
            <h:commandLink value="BACK" action="#{page.viewObjects}"
              styleClass="link_back" /></div></h:form></f:view></ui:define>
</ui:composition></html>
```

/admin/object/index.jsp

```
<jsp:forward page="view.jsf" />
```

/admin/object/object.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:a4j="http://richfaces.org/a4j"
   xmlns:rich="http://richfaces.org/rich">
<ui:composition template="../includes/template.xhtml">
   <ui:define name="title">View Object</ui:define>
   <ui:define name="content"><f:view><h:form>
            <rich:panel headerClass="view_header" styleClass="panel_main"
id="datatable">
              <f:facet name="header"><h:outputText value="Object Details" />
              </f:facet>
              <h:panelGrid styleClass="panel_details" >
              <h:panelGrid columns="2">
                <h:outputText value="Class name: "  styleClass="text_label"/>
                <h:outputText value="#{object.name}" />
                <h:outputText value="Description:"  styleClass="text_label"/>
                <h:outputText value="#{object.description}" />
              </h:panelGrid></h:panelGrid>
              <h:panelGrid styleClass="panel_contents">
                <rich:dataTable value="#{object.attrs}" var="attr"
                  id="output_attribute" headerClass="view_subheader"
                  styleClass="panel_datatable" footerClass="view_footer"
                  rendered="#{!object.editAttr}">
                  <f:facet name="header"><h:outputText value="Attributes" />
                  </f:facet>
                  <rich:column width="25%">
                    <f:facet name="header">
                      <h:outputText value="Name" /></f:facet>
                    <h:outputText value="#{attr.name}"
                      rendered="#{!attr.removed}" /></rich:column>
          <rich:column width="25%">
           <f:facet name="header"> <h:outputText value="Variable" />
</f:facet>  <h:outputText value="#{attr.variable}"
rendered="#{!attr.removed}" />     </rich:column>
                  <rich:column width="20%">
                    <f:facet name="header">
                      <h:outputText value="Data Type" /></f:facet>
                    <h:outputText value="#{attr.stringDatatype}"
                      rendered="#{!attr.removed}" /></rich:column>
                  <rich:column width="30%">
                    <f:facet name="header">
                      <h:outputText value="Description" /></f:facet>
                    <h:outputText value="#{attr.description}"
                      rendered="#{!attr.removed}" /></rich:column>
                  <f:facet name="footer">
                    <a4j:commandButton value="Update Object"
                      action="#{object.editData }">
                      <rich:toolTip value="Make changes to this object"
styleClass="tooltip"/></a4j:commandButton></f:facet></rich:dataTable></h:pan
elGrid></rich:panel>
            <div id="link_back"><br />
            <h:commandLink value="BACK" action="#{page.viewObjects}"
              styleClass="link_back" /></div>
          </h:form></f:view></ui:define></ui:composition></html>
```

/admin/object/view.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:a4j="http://richfaces.org/a4j"
   xmlns:rich="http://richfaces.org/rich">
<ui:composition template="../includes/template.xhtml">
   <ui:define name="title">Object Manager</ui:define>
   <ui:define name="content"><f:view><h:form>
            <rich:dataTable value="#{table.objects}" var="obj"
              styleClass="view_datatable" headerClass="view_header"
              footerClass="view_footer">
              <f:facet name="header">
                <h:outputText value="Objects" /></f:facet>
              <rich:column headerClass="delete_columnHeader">
                <f:facet name="header">
                  <h:commandButton action="#{table.deleteObjects}"
                    onclick="if (!confirm('#{table.askDelObj}')) return false"
                    value="delete" /></f:facet>
                <h:selectBooleanCheckbox value="#{obj.removed}" />
              </rich:column>
              <rich:column headerClass="view_subheader"
sortBy="#{obj.name}">
                <f:facet name="header">
                  <h:outputText value="Name" /></f:facet>
                <h:commandLink value="#{obj.name}"
action="#{link.viewObject}">
                  <f:param name="object" value="#{obj.id}" />
                </h:commandLink></rich:column>
              <rich:column headerClass="view_subheader"
sortBy="#{obj.description}">
                <f:facet name="header">
                  <h:outputText value="Description" /></f:facet>
                <h:outputText value="#{obj.description}" /></rich:column>
      <rich:column headerClass="view_subheader"
sortBy="#{obj.created_by}">
        <f:facet name="header">
          <h:outputText value="Created by" /></f:facet>
        <h:outputText value="#{obj.created_by}" /></rich:column>
      <rich:column headerClass="view_subheader"
sortBy="#{obj.date_created}">
        <f:facet name="header">
          <h:outputText value="Date created" />
        </f:facet>
        <h:outputText value="#{obj.date_created}" />
      </rich:column>
      <rich:column headerClass="view_subheader"
sortBy="#{obj.modified_by}">
        <f:facet name="header">
          <h:outputText value="Modified by" />
        </f:facet><h:outputText value="#{obj.modified_by}" /></rich:column>
      <rich:column headerClass="view_subheader">
```

```
        <f:facet name="header">
          <h:outputText value="Date last modified"
sortBy="#{obj.date_modified}"/>      </f:facet>
         <h:outputText value="#{obj.date_modified}" />
        </rich:column>
              <f:facet name="footer">
                <h:commandButton action="#{page.addObject}"   value="Create
new object" />
              </f:facet></rich:dataTable></h:form></f:view></ui:define>
</ui:composition></html>
```

/admin/rule/add.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:a4j="http://richfaces.org/a4j"
   xmlns:rich="http://richfaces.org/rich">
<ui:composition template="../includes/template.xhtml">
   <ui:define name="title"><h:outputText value="#{ruleset.title}" /></ui:define>
   <ui:define name="content"><f:view><h:form>
            <rich:panel styleClass="panel_main" headerClass="view_header"
id="panel">
              <f:facet name="header">
                <h:outputText value="Ruleset Details" />  </f:facet>
<!-- RULESET DETAILS -->
              <h:panelGrid columns="2" styleClass="panel_details"  >
              <h:outputText value="Ruleset name: " styleClass="text_label" />
                <h:inputText value="#{ruleset.name}"
styleClass="textbox_name"/>
                <h:outputText value="Description:" styleClass="text_label" />
                <h:inputTextarea value="#{ruleset.description}"
                   styleClass="textarea_desc" />
               <h:outputText value="Rule Group:"  styleClass="text_label" />
                <h:selectOneMenu value="#{ruleset.rulegroup}" >
          <f:selectItems value="#{ruleset.selectRulegroups}" />
        </h:selectOneMenu>
                <h:outputText value="Type:" styleClass="text_label" />
        <h:selectOneRadio value="#{ruleset.type}" disabled="#{ruleset.set}">
          <f:selectItem itemValue="1" itemLabel="Non-sequential" />
          <f:selectItem itemValue="2" itemLabel="Sequential"/>
          <a4j:support event="onchange" >
            <rich:toolTip value="tooltip :)" />
          </a4j:support>
        </h:selectOneRadio>
        <h:outputText value="Objects to use:" styleClass="text_label" />
        <h:selectManyListbox value="#{ruleset.arrObjects}"
rendered="#{!ruleset.set}">
          <f:selectItems value="#{ruleset.allObjects}"/>
        </h:selectManyListbox>
        <h:outputText value="" rendered="#{ruleset.set}" />
        <h:outputText value="'No Result' message: " styleClass="text_label" />
        <h:inputText value="#{ruleset.conclusion}" styleClass="textbox_name"
/>
        <a4j:commandButton value="Start" reRender="panel"
           action="#{ruleset.typeSet}" rendered="#{!ruleset.set}">
           <rich:toolTip value="Start writing the rules for this ruleset"
styleClass="tooltip" />
          </a4j:commandButton> </h:panelGrid>
              <h:panelGrid styleClass="panel_contents"
rendered="#{ruleset.set}">
                <rich:dataTable id="datatable" value="#{ruleset.arrRules}"
                   var="rule"  headerClass="view_subheader"
          styleClass="panel_datatable" footerClass="view_footer">
                   <f:facet name="header">
                      <h:outputText value="Rules" />
                   </f:facet>
                   <rich:column styleClass="method_column">
<!-- RULE DETAILS -->
             <a4j:commandButton value="x" reRender="datatable"
 styleClass="button_delete"  action="#{rule.removeThis}"onclick="if
(!confirm('#{rule.askDel}')) return false" >
                <rich:toolTip value="Remove rule" styleClass="tooltip"/>
              </a4j:commandButton>
              <h:outputText value="Rule #{rule.ruleId}"
styleClass="text_RuleId"/><br /><br />
              <h:outputText value="Priority #:" styleClass="text_label"/>
              <h:inputText value="#{rule.salience}" style="width:30px;"/><br />
              <h:selectBooleanCheckbox value="#{rule.no_loop}"/>
              <h:outputText value="no-loop" styleClass="text_label"/><br />
              <h:outputText value="Group #:" styleClass="text_label"
rendered="#{!rule.start &amp;&amp; ruleset.type==2}"/>
              <h:inputText value="#{rule.agenda}" style="width:30px;"
rendered="#{!rule.start &amp;&amp; ruleset.type==2}">
                <a4j:support event="onchange" reRender="datatable"
action="#{ruleset.initSelectAgenda}" />
              </h:inputText><br />
              <h:selectBooleanCheckbox value="#{rule.start}"
rendered="#{ruleset.type==2}">
                <a4j:support event="onchange" reRender="datatable" />
              </h:selectBooleanCheckbox>
              <h:outputText value="Start rule" styleClass="text_label"
rendered="#{ruleset.type==2  </rich:column>
          <rich:column><!-- IF PART -->
                    <h:panelGrid rendered="#{!rule.removed}">
                    <table><tr><td><h:panelGrid columns="2">
                      <h:outputText value="IF" />
                  <rich:panel styleClass="panel_rule">
                    <table width="100%">
                      <ui:repeat value="${rule.arrCond}" var="condition">
                    <tr> <td> <h:selectOneMenu value="#{condition.bool}"
rendered="#{rule.showBool &amp;&amp; condition.bool!=-1
&amp;&amp; !condition.removed}">
                       <f:selectItem itemValue="1" itemLabel="and" />
                       <f:selectItem itemValue="0" itemLabel="or" />
                      </h:selectOneMenu>  </td>
                      <td><h:selectOneMenu value="#{condition.objId}"
rendered="#{condition.showObjAttr &amp;&amp; !condition.removed}">
                       <f:selectItems value="#{ruleset.selectObjects}" />
                       <a4j:support event="onchange" reRender="datatable" />
                      </h:selectOneMenu>
                      <h:selectOneMenu value="#{condition.attrId}"
rendered="#{condition.showObjAttr &amp;&amp; !condition.removed}">
                       <f:selectItems value="#{condition.attributes}" />
                      </h:selectOneMenu>
                     <h:selectOneMenu value="#{condition.function}"
                   rendered="#{condition.showFuncLeft
&amp;&amp; !condition.removed}">
                      <f:selectItems value="#{ruleset.selectFunctions}" />
                     </h:selectOneMenu>
                       <h:selectOneMenu value="#{condition.optrId}"
rendered="#{!condition.removed}">
                        <f:selectItems value="#{ruleset.selectOperators}" />
                        <a4j:support event="onchange" reRender="datatable" />
                       </h:selectOneMenu>
                       <h:inputText value="#{condition.value}"
rendered="#{condition.showInputText &amp;&amp; !condition.removed}"  />
                       <h:selectOneMenu value="#{condition.value}"
rendered="#{condition.showFuncRight &amp;&amp; !condition.removed}">
                     <f:selectItems value="#{ruleset.selectFunctions}" />
                    </h:selectOneMenu>
                    <h:outputText value="from" styleClass="text_label"
rendered="#{condition.showToFrom &amp;&amp; !condition.removed}"/>
                     <h:inputText value="#{condition.from}" style="width:30px;"
rendered="#{condition.showToFrom &amp;&amp; !condition.removed}" />
                    <h:outputText value="to" styleClass="text_label"
rendered="#{condition.showToFrom &amp;&amp; !condition.removed}"/>
                      <h:inputText value="#{condition.to}" style="width:30px;"
rendered="#{condition.showToFrom &amp;&amp; !condition.removed}"
/></td><td>
```

```
                <h:selectOneMenu value="#{condition.type}"
rendered="#{!condition.removed}">
                 <f:selectItem itemValue="1" itemLabel="Attribute-Value" />
                 <f:selectItem itemValue="2" itemLabel="Attribute-Function" />
                 <f:selectItem itemValue="3" itemLabel="Function-Function" />
                 <f:selectItem itemValue="4" itemLabel="Function-Value" />
                 <a4j:support event="onchange" reRender="datatable" />
                </h:selectOneMenu>
                  <a4j:commandButton value="x" reRender="datatable"
styleClass="button_delete"  rendered="#{rule.showBool &amp;&amp;
condition.bool!=-1 &amp;&amp; !condition.removed}"
action="#{condition.removeThis}"  onclick="if (!confirm('#{rule.askCond}'))
return false">
                   <rich:toolTip value="Remove condition"
styleClass="tooltip"/>
                  </a4j:commandButton></td> </tr> </ui:repeat>
                <tr><td colspan="3">
                  <a4j:commandButton value="Add Condition"
reRender="datatable"action="#{rule.addCondition}" >
                    <rich:toolTip value="Adds a new condition for this rule"
styleClass="tooltip"/>
                   </a4j:commandButton></td></tr></table> </rich:panel>
</h:panelGrid></td></tr><!-- THEN PART -->
            <tr><td><h:panelGrid columns="2">
               <h:outputText value="THEN" />
              <rich:panel styleClass="panel_rule">
                <table width="100%">
                <ui:repeat value="${rule.arrCons}" var="consequence">
                <tr><td><h:selectOneMenu value="#{consequence.type}">
                    <f:selectItems value="#{ruleset.conseqTypes}" />
                    <a4j:support event="onchange" reRender="datatable" />
                   </h:selectOneMenu> </td>
                <td> <h:selectOneMenu value="#{consequence.objId}"
rendered="#{consequence.showObjectAttr}">
                    <f:selectItems value="#{ruleset.selectObjects}" />
                    <a4j:support event="onchange" reRender="datatable" />
                   </h:selectOneMenu>
                   <h:selectOneMenu value="#{consequence.attrId}"
rendered="#{consequence.showObjectAttr}">
                    <f:selectItems value="#{consequence.attributes}" />
                   </h:selectOneMenu>
                 <h:outputText value=" = " rendered="#{consequence.modify }" />
                   <h:inputText value="#{consequence.value}"
                    rendered="#{consequence.showInputText}" />
                    <h:selectOneMenu value="#{consequence.function}"
rendered="#{consequence.showMethods}">
                    <f:selectItems value="#{ruleset.selectFunctions}" />
                   </h:selectOneMenu>
                  <h:selectOneMenu value="#{consequence.value}"
rendered="#{consequence.showRules}">
                    <f:selectItems value="#{ruleset.selectRules}" />
                  </h:selectOneMenu>
                    <h:selectOneMenu rendered="#{consequence.modify ||
consequence.decision}" value="#{consequence.valOrFunc }">
                  <f:selectItem itemValue="1" itemLabel="Call function"/>
                  <f:selectItem itemValue="0" itemLabel="Use custom value"/>
                   <a4j:support event="onchange"
action="#{consequence.onChange }"  reRender="datatable" />
                   </h:selectOneMenu> </td> </tr> </ui:repeat>
                 <tr><td colspan="2"><a4j:commandButton value="Add
Consequence" reRender="datatable"  action="#{rule.addConsequence}" >
                    <rich:toolTip value="Adds a new consequence to this rule"
styleClass="tooltip"/></a4j:commandButton>
                   <a4j:commandButton value="Remove Last"
reRender="datatable"  rendered="#{rule.allowRemove}" onclick="if
(!confirm('#{rule.askCons}')) return false" action="#{rule.removeConsequence}"
>
                    <rich:toolTip value="Removes the last consequence from the
list" styleClass="tooltip"/> </a4j:commandButton> </td> </tr> </table>
</rich:panel> </h:panelGrid></td></tr> </table></h:panelGrid></rich:column>
                  <f:facet name="footer">
                    <h:panelGrid columns="3">
                    <h:outputText value="Group #:" styleClass="text_label"
rendered="#{ruleset.type==2}" />
                    <h:inputText value="#{ruleset.agenda}"
rendered="#{ruleset.type==2}" styleClass="textbox_number"/>
              <a4j:commandButton value="Add Rule" reRender="datatable"
action="#{ruleset.addRule}">
                <rich:toolTip value="Adds a new rule for this ruleset"
styleClass="tooltip"/></a4j:commandButton></h:panelGrid> </f:facet>
</rich:dataTable><!-- RULESET FUNCTION -->
          <rich:dataTable id="function_datatable" value="#{ruleset.arrFunctions}"
 var="function"  headerClass="view_subheader" styleClass="panel_datatable"
footerClass="view_footer">
     <f:facet name="header"> <h:outputText value="Functions" /></f:facet>
            <rich:column>
             <h:panelGrid columns="2" rendered="#{!function.removed}">
                <h:panelGrid>
                 <h:panelGrid columns="2">
                   <h:outputText value="Function name: "
styleClass="text_label"/>
                    <h:inputText value="#{function.name}"
styleClass="textbox_name" disabled="true"/>
                   <h:outputText value="Description: " styleClass="text_label"/>
                   <h:inputText value="#{function.description}"
styleClass="textbox_desc"/>
                   <h:outputText value="Return type: " styleClass="text_label"/>
                   <h:selectOneMenu value="#{function.returntype}">
                         <f:selectItems value="#{ruleset.selectDataTypes}" />
                      </h:selectOneMenu>    </h:panelGrid>
                  <h:outputText value="Parameters :" />
                  <h:dataTable value="#{function.params}" var="parameter" >
                   <h:column><f:facet name="header">
                      <h:outputText value="Variable Name"
styleClass="text_label" /></f:facet>
                     <h:inputText value="#{parameter.variable}"
rendered="#{!parameter.removed}"/> </h:column>
                    <h:column><f:facet name="header">
                      <h:outputText value="Input type" styleClass="text_label" />
                    </f:facet>
                     <h:selectOneMenu value="#{parameter.objId}"
rendered="#{!parameter.removed}">
                      <f:selectItems value="#{ruleset.selectObjects}" />
                      <a4j:support event="onchange" reRender="datatable" />
                     </h:selectOneMenu>
                     <h:selectOneMenu value="#{parameter.attrId}"
rendered="#{!parameter.removed}">
                      <f:selectItems value="#{parameter.attributes}" />
                     </h:selectOneMenu></h:column>
                   <f:facet name="footer">
                     <a4j:commandButton value="Add"
                      action="#{function.addParameter }"
reRender="function_datatable"><rich:toolTip value="Add a parameter"
styleClass="tooltip" /> </a4j:commandButton> </f:facet> </h:dataTable>
</h:panelGrid> <h:panelGrid>
                <a4j:commandButton value="Remove Function"
                 action="#{function.removeThis}" reRender="function_datatable,
datatable"onclick="if (!confirm('#{function.askDel}')) return false">
                  <rich:toolTip value="Removes this function" styleClass="tooltip"
/></a4j:commandButton>
                <h:outputText value="Function code:" styleClass="text_label"/>
                <h:inputTextarea value="#{function.code}"
styleClass="textArea_code"></h:inputTextarea </h:panelGrid> </h:panelGrid>
</rich:column>
            <f:facet name="footer">
                <h:panelGrid columns="3">
                  <h:outputText value="New function name:"
styleClass="text_label"/>
                  <h:inputText value="#{ruleset.newFuncName}"
styleClass="text_name" />
```

```
            <a4j:commandButton value="Add" reRender="function_datatable,
datatable"
                action="#{ruleset.addFunction}">
                 <rich:toolTip value="Adds a new function for this ruleset"
styleClass="tooltip"/> </a4j:commandButton></h:panelGrid></f:facet>
        </rich:dataTable><!-- END --> </h:panelGrid>
        <h:panelGrid styleClass="panel_footer" rendered="#{ruleset.set}">
            <h:commandButton action="#{ruleset.save}" value="Save Ruleset"
rendered="#{!ruleset.edit}"/>
       <h:commandButton action="#{ruleset.update}" value="Save Changes"
rendered="#{ruleset.edit}"/></h:panelGrid></rich:panel>
        <div id="link_back"><br />
        <h:commandLink value="BACK" action="#{page.viewRules}"
            styleClass="link_back" /></div></h:form></f:view></ui:define>
</ui:composition></html>
```

/admin/rule/form.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:a4j="http://richfaces.org/a4j"
   xmlns:rich="http://richfaces.org/rich">
<ui:composition template="../includes/template.xhtml">
   <ui:define name="title">Input Forms</ui:define>
   <ui:define name="content"><f:view><h:form>
           <rich:panel headerClass="view_header" styleClass="panel_main">
              <f:facet name="header">
                 <h:outputText value="Input Forms" />
              </f:facet>
              <h:panelGrid columns="2" styleClass="panel_details">
                 <h:outputText value="Ruleset name: " styleClass="text_label"
/><h:outputText value="#{ruleset.name}" /></h:panelGrid>
   <h:panelGrid styleClass="panel_contents">
                 <rich:dataTable value="#{ruleset.arrIForms}" var="form"
id="datatable" headerClass="view_subheader" styleClass="panel_datatable"
           footerClass="view_footer" >
                    <f:facet name="header">
                       <h:outputText value="attributes" />    </f:facet>
          <rich:column>
           <f:facet name="header">
             <h:outputText value="Required" /> </f:facet>
           <h:selectBooleanCheckbox value="#{form.status}"/>
          </rich:column><rich:column>
                       <f:facet name="header">
                          <h:outputText value="attribute" />                  </f:facet>
   <h:outputText value="#{form.stringAttribute}" />
   </rich:column>
                    <rich:column>
                       <f:facet name="header">
                          <h:outputText value="question" /></f:facet>
                       <h:inputText value="#{form.question}"
styleClass="textbox_form"/>  </rich:column>
                    <rich:column>
                       <f:facet name="header">
                          <h:outputText value="input type" /></f:facet>
                       <h:selectOneMenu value="#{form.inputType}">
                          <f:selectItems value="#{ruleset.selectInputTypes}" />
                          <a4j:support event="onchange" reRender="datatable"
/></h:selectOneMenu></rich:column>
                    <rich:column>
                       <f:facet name="header">
                          <h:outputText value="choices (separated by ';' )" />
                       </f:facet>
                       <h:inputText value="#{form.choices}"
rendered="#{!form.inputText}"    styleClass="textbox_form" /> </rich:column>
                    <f:facet name="footer">
                       <h:commandButton action="#{ruleset.updateForm}"
   value="Save Changes" />  </f:facet></rich:dataTable></h:panelGrid>
           </rich:panel><div id="link_back"><br />
           <h:commandLink value="BACK" action="#{page.viewRules}"
               styleClass="link_back" /></div></h:form>  </f:view>
</ui:define></ui:composition></html>
```

/admin/rule/group.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:a4j="http://richfaces.org/a4j"
   xmlns:rich="http://richfaces.org/rich">
<ui:composition template="../includes/template.xhtml">
   <ui:define name="title">Manage Rulegroups</ui:define>
   <ui:define name="content"><f:view>  <h:form>
           <rich:dataTable value="#{table.rulegroups}" var="rg"
              styleClass="view_datatable" headerClass="view_header"
       footerClass="view_footer" >
              <f:facet name="header"><h:outputText value="Rulegoups" />
              </f:facet>
              <rich:column headerClass="delete_columnHeader">
                 <f:facet name="header">
                    <h:commandButton action="#{table.deleteRulegroups}"
                       onclick="if (!confirm('#{table.askDelRg}')) return false"
                       value="Delete" />    </f:facet>
                 <h:selectBooleanCheckbox value="#{rg.removed}"
rendered="#{!rg.default}" /></rich:column>
              <rich:column headerClass="view_subheader"
sortBy="#{rg.name}"> <f:facet name="header">
                    <h:outputText value="Name" />     </f:facet>
                 <h:outputText value="#{rg.name}" /> </rich:column>
              <rich:column headerClass="view_subheader"
sortBy="#{rg.description}"><f:facet name="header">
                    <h:outputText value="Description" /> </f:facet>
                 <h:outputText value="#{rg.description}" /> </rich:column>
        <rich:column headerClass="view_subheader"
sortBy="#{rg.created_by}">   <f:facet name="header">
           <h:outputText value="Created by" />  </f:facet>
         <h:outputText value="#{rg.created_by}" /> </rich:column>
        <rich:column headerClass="view_subheader"
sortBy="#{rg.date_created}">  <f:facet name="header">
           <h:outputText value="Date created" />  </f:facet>
         <h:outputText value="#{rg.date_created}" />  </rich:column>
              <f:facet name="footer">
                <h:panelGrid columns="5">
                 <h:outputText value="New rulegroup name: "
styleClass="text_label"/>
                 <h:inputText value="#{table.newRg.name}" />
                 <h:outputText value="Description: " styleClass="text_label"/>
           <h:inputText value="#{table.newRg.description}" />
                 <a4j:commandButton action="#{table.saveNewRg }"
value="Add">
                  <rich:toolTip value="Add new rule group" styleClass="tooltip"/>
</a4j:commandButton></h:panelGrid></f:facet>
</rich:dataTable></h:form></f:view></ui:define></ui:composition></html>
```

/admin/rule/index.jsp

```
<jsp:forward page="view.jsf" />
```

/admin/rule/rule.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:f="http://java.sun.com/jsf/core"
  xmlns:a4j="http://richfaces.org/a4j"
  xmlns:rich="http://richfaces.org/rich">
<ui:composition template="../includes/template.xhtml">
  <ui:define name="title">View Ruleset</ui:define>
  <ui:define name="content">   <f:view>     <h:form>
       <rich:panel styleClass="panel_main" headerClass="view_header"
id="panel">
        <f:facet name="header"> <h:outputText value="Ruleset Details" />
        </f:facet>
<!-- RULESET DETAILS -->
```

```
        <h:panelGrid columns="2" styleClass="panel_details">
          <h:outputText value="Ruleset name: " styleClass="text_label" />
          <h:outputText value="#{ruleset.name}" styleClass="textbox_name" />
          <h:outputText value="Description:" styleClass="text_label" />
          <h:outputText value="#{ruleset.description}"/>
          <h:outputText value="Rule Group:"  styleClass="text_label" />
          <h:outputText value="#{ruleset.stringRulegroup}" />
          <h:outputText value="Type:" styleClass="text_label" />
          <h:outputText value="#{ruleset.stringType}"/>
          <h:outputText value="'No Result' message: " styleClass="text_label" />
<h:outputText value="#{ruleset.conclusion}"/>
        </h:panelGrid>\
        <h:panelGrid styleClass="panel_contents">\
          <rich:dataTable value="#{ruleset.arrRules}" var="rule"
            id="datatable1" headerClass="view_subheader"
            styleClass="panel_datatable" footerClass="view_footer" >\
            <f:facet name="header">
              <h:outputText value="Rules" />
            </f:facet>
            <rich:column styleClass="method_column">\
<!-- RULE DETAILS -->
              <h:outputText value="Rule #{rule.ruleId}" styleClass="text_RuleId"
/><br /><br />
              <h:outputText value="Priority #:  " styleClass="text_label"/>
              <h:outputText value="#{rule.salience}" style="width:30px;"/>
<br/> <h:selectBooleanCheckbox value="#{rule.no_loop}" disabled="true"/>
                <h:outputText value="No-loop" styleClass="text_label"/>
<br /><h:outputText value="Group #:  " styleClass="text_label"
rendered="#{ruleset.type==2}"/>
                <h:outputText value="#{rule.agenda}" style="width:30px;"
rendered="#{ruleset.type==2}"/><br />
                <h:selectBooleanCheckbox value="#{rule.start}"
rendered="#{ruleset.type==2}" disabled="true"/>
                <h:outputText value="Start rule" styleClass="text_label"
rendered="#{ruleset.type==2}"/> </rich:column>  <rich:column>
<!-- IF PART -->
              <h:panelGrid><table> <tr> <td> <h:panelGrid columns="2">
                <h:outputText value="IF" styleClass="panel_ifThen" />
                  <rich:panel styleClass="panel_rule">
                    <ui:repeat value="${rule.arrCond}" var="condition">
                      <h:outputText value="#{condition.stringObject}."
rendered="#{condition.showObjAttr}"/>
                      <h:outputText value="#{condition.stringAttribute} "
rendered="#{condition.showObjAttr}"/>
                      <h:outputText value="#{condition.function} "
rendered="#{!condition.showObjAttr}" />
                      <h:outputText value="#{condition.stringOperator}  " />
                      <h:outputText value="#{condition.value}" />
                     <br/></ui:repeat></rich:panel></h:panelGrid></td></tr>
<!-- THEN PART -->
              <tr><td> <h:panelGrid columns="2">
                <h:outputText value="THEN" styleClass="panel_ifThen" />
                  <rich:panel styleClass="panel_rule">
                    <ui:repeat value="${rule.arrCons}" var="consequence">
                      <h:outputText value="#{consequence.stringConsequence} :"
/><h:outputText value="#{consequence.stringObject}'s "
rendered="#{consequence.showObjectAttr}"/>
                      <h:outputText value="#{consequence.stringAttribute} "
rendered="#{consequence.showObjectAttr}"/>
                      <h:outputText value=" = " rendered="#{consequence.modify}"
/>
                    <h:outputText value=" group "
rendered="#{consequence.showRules}" />
                      <h:outputText value="#{consequence.value}"
rendered="#{consequence.showInputText || consequence.showRules}"/>
                    <h:outputText value="#{consequence.function}()"
rendered="#{consequence.showMethods}" /> <br /></ui:repeat>
                </rich:panel></h:panelGrid></td> </tr></table>
              </h:panelGrid> </rich:column></rich:dataTable>
<!-- FUNCTIONS -->
        <rich:dataTable value="#{ruleset.arrFunctions}" var="function"
          id="function_datatable" headerClass="view_subheader"
          styleClass="panel_datatable" footerClass="view_footer" >
          <f:facet name="header"> <h:outputText value="Functions" />
          </f:facet> <rich:column>
            <h:panelGrid columns="2" rendered="#{!function.removed}">
             <h:panelGrid><h:panelGrid columns="2">
                <h:outputText value="Function name: " styleClass="text_label"/>
                <h:outputText value="#{function.name}" />
                <h:outputText value="Description: " styleClass="text_label"/>
                <h:outputText value="#{function.description}" />
                <h:outputText value="Return type: " styleClass="text_label"/>
                <h:outputText value="#{function.stringDatatype}" />
              </h:panelGrid> <h:outputText value="Parameters :" /> <br />
              <ui:repeat value="${function.params}" var="parameter">
                <h:outputText value="#{parameter.variable} = " />
                <h:outputText value="#{parameter.stringObj}'s "/>
                <h:outputText value="#{parameter.stringAtt}"/>
                <h:outputText value=" = " rendered="#{consequence.modify }"
/><br /></ui:repeat></h:panelGrid>
              <h:panelGrid>
                <h:outputText value="Code: " styleClass="text_label"/>
                <h:outputText value="#{function.code}" /> </h:panelGrid>
            </h:panelGrid></rich:column></rich:dataTable>
        <h:panelGrid styleClass="panel_footer">
          <a4j:commandButton value="Update ruleset"
action="#{ruleset.editData}">
            <rich:toolTip value="Make changes in this ruleset"
styleClass="tooltip"/> </a4j:commandButton> </h:panelGrid>
</h:panelGrid></rich:panel>     <div id="link_back"><br />
       <h:commandLink value="BACK" action="#{page.viewRules}"
         styleClass="link_back" /></div> </h:form> </f:view></ui:define>
</ui:composition></html>
```

/admin/rule/view.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:a4j="http://richfaces.org/a4j"
   xmlns:rich="http://richfaces.org/rich">
<ui:composition template="../includes/template.xhtml">
   <ui:define name="title">Manage Rules</ui:define>
   <ui:define name="content">      <f:view>         <h:form>
      <a4j:commandLink action="#{page.viewRulegroups}"  value="Manage
Rule Groups" style="text-align:right;">
        <rich:toolTip value="Open page for updating view rule groups."
styleClass="tooltip"/>       </a4j:commandLink>       <br />
            <rich:dataTable value="#{table.rulesets}" var="rs"
              styleClass="view_datatable" headerClass="view_header"
       footerClass="view_footer">
              <f:facet name="header">   <h:outputText value="Rulesets" />
              </f:facet>
              <rich:column headerClass="delete_columnHeader">
                <f:facet name="header">
                  <h:commandButton action="#{table.deleteRulesets}"
                    onclick="if (!confirm('#{table.askDelRul}')) return false"
                    value="Delete" />
                </f:facet>
                <h:selectBooleanCheckbox value="#{rs.removed}" />
              </rich:column>
              <rich:column headerClass="view_subheader"
sortBy="#{rs.name}">
                <f:facet name="header">   <h:outputText value="Name" />
                </f:facet> <h:commandLink value="#{rs.name}"
action="#{link.viewRule}">  <f:param name="ruleset" value="#{rs.id}" />
                </h:commandLink>     </rich:column>
              <rich:column headerClass="view_subheader"
sortBy="#{rs.description}"> <f:facet name="header">
                  <h:outputText value="Description" /> </f:facet>
                <h:outputText value="#{rs.description}" /> </rich:column>
```

```
        <rich:column headerClass="view_subheader"
sortBy="#{rs.stringRulegroup}">     <f:facet name="header">
            <h:outputText value="Rule group" />   </f:facet>
          <h:outputText value="#{rs.stringRulegroup}" />     </rich:column>
        <rich:column headerClass="view_subheader"
sortBy="#{rs.created_by}">
          <f:facet name="header">  <h:outputText value="Created by" />
</f:facet><h:outputText value="#{rs.created_by}" /></rich:column>
        <rich:column headerClass="view_subheader"
sortBy="#{rs.date_created}">
          <f:facet name="header">
            <h:outputText value="Date created" /></f:facet>
          <h:outputText value="#{rs.date_created}" /></rich:column>
        <rich:column headerClass="view_subheader"
sortBy="#{rs.modified_by}">
          <f:facet name="header">
            <h:outputText value="Modified by" /></f:facet>
          <h:outputText value="#{rs.modified_by}" /></rich:column>
        <rich:column headerClass="view_subheader">
          <f:facet name="header">
            <h:outputText value="Date last modified" sortBy="#{rs.date_modified}"
/></f:facet>
          <h:outputText value="#{rs.date_modified}" /></rich:column>
              <rich:column headerClass="view_subheader"
sortBy="#{rs.formSetString}">
                  <f:facet name="header">
                     <h:outputText value="Input Form" /></f:facet>
                  <h:commandLink value="#{rs.formSetString}"
                     action="#{link.viewForm}">
                     <f:param name="ruleset" value="#{rs.id}" />
                  </h:commandLink> </rich:column>
               <rich:column headerClass="view_subheader" id="status"
sortBy="#{rs.status}">
                  <f:facet name="header">
                     <h:outputText value="Status" /> </f:facet>
          <h:selectBooleanCheckbox value="#{rs.status}"
             onclick="if (!confirm('#{rs.askDeploy}')) return false" >
           <a4j:support event="onchange" action="#{rs.publish}"
reRender="status, hMessage">
              <rich:toolTip styleClass="tooltip" value="Publish/Unpublish this
ruleset"/></a4j:support> </h:selectBooleanCheckbox></rich:column>
                <f:facet name="footer">
                   <h:commandButton action="#{page.addRuleset}"
                      value="Create new rule" />
                </f:facet></rich:dataTable></h:form></f:view></ui:define>
</ui:composition></html>
```

/admin/setting/index.jsp

```
<jsp:forward page="view.jsf" />
```

/admin/rule/setting/templates.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:a4j="http://richfaces.org/a4j"
   xmlns:rich="http://richfaces.org/rich"
   xmlns:t="http://myfaces.apache.org/tomahawk">
<ui:composition template="../includes/template.xhtml">
   <ui:define name="title">Project Template</ui:define>
   <ui:define name="content">      <f:view>         <h:form>
          <rich:dataTable value="#{table.templates}" var="template"
              styleClass="view_datatable" headerClass="view_header"
        footerClass="view_footer" id="datatable">
             <f:facet name="header">
                <h:outputText value="Templates" /></f:facet>
             <rich:column headerClass="delete_columnHeader">
                <f:facet name="header">
                  <a4j:commandButton action="#{table.deleteTemplates}"
          onclick="if (!confirm('#{table.askDelTem}')) return false"
          value="Delete">
                     <rich:toolTip styleClass="tooltip" value="Delete checked
templates" /></a4j:commandButton></f:facet>
             <h:selectBooleanCheckbox value="#{template.removed}"
rendered="#{!template.default &amp;&amp; !template.used}"/></rich:column>
               <rich:column headerClass="view_subheader"
sortBy="#{template.name}">
                  <f:facet name="header">
                     <h:outputText value="Template name" /></f:facet>
                  <h:outputText value="#{template.name}" /></rich:column>
               <rich:column headerClass="view_subheader"
sortBy="#{template.username}">
                  <f:facet name="header">
                    <h:outputText value="Uploaded by" />      </f:facet>
                  <h:outputText value="#{template.username}" />      </rich:column>
               <rich:column headerClass="view_subheader"
sortBy="#{template.date_uploaded}">
                  <f:facet name="header">
                    <h:outputText value="Date_uploaded" />        </f:facet>
                  <h:outputText value="#{template.date_uploaded}" />    </rich:column>
               <rich:column headerClass="view_subheader"
sortBy="#{template.used}">
                  <f:facet name="header">
                    <h:outputText value="Status" />   </f:facet>
                  <a4j:commandLink value="Use" action="#{template.useThis}"
rendered="#{!template.used}"
                    onclick="if (!confirm('Use this template for the front-end part of the
project?')) return false"   reRender="datatable">
                    <rich:toolTip styleClass="tooltip" value="Set this as the default
template to be used"/></a4j:commandLink>
                  <h:outputText value="Currently used" rendered="#{template.used}" />
                </rich:column></rich:dataTable><br />
               <rich:fileUpload id="upload"
fileUploadListener="#{template.fileUploadListener}"
                    immediateUpload="true" listHeight="60"  acceptedTypes="css, zip"
autoclear="true"  noDuplicate="true" disabled="false">
                    <a4j:support event="onfileuploadcomplete" reRender="datatable,
hMessage" /> <a4j:support event="onerror" reRender="hMessage" />
                </rich:fileUpload><br /><br />
             <div id="link_back"><br />
               <a4j:commandLink  value="BACK" action="#{page.viewSettings}"
             styleClass="link_back">
                 <rich:toolTip styleClass="tooltip" value="Go back to"></rich:toolTip>
               </a4j:commandLink><h:commandLink /></div></h:form></f:view>
          </ui:define></ui:composition></html>
```

/admin/setting/view.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:a4j="http://richfaces.org/a4j"
    xmlns:rich="http://richfaces.org/rich">
<ui:composition template="../includes/template.xhtml">
  <ui:define name="title">Project Settings</ui:define>
  <ui:define name="content"> <f:view><h:form>
     <rich:panel  headerClass="view_header" styleClass="panel_main">
       <f:facet name="header" >
          <h:outputText value="Configuration Details" />  </f:facet>
         <h:panelGrid columns="2" styleClass="panel_details"
         headerClass="view_subheader" footerClass="view_footer">
            <f:facet name="header">
              <h:outputText value="Project Settings" /> </f:facet>
              <h:outputText value="Site name :"  styleClass="text_label"/>
           <h:inputText value="#{site.name}" styleClass="textbox_form"/>
           <h:outputText value="Homepage URL : "  styleClass="text_label"/>
           <h:inputText value="#{site.url}" styleClass="textbox_form"/>
           <h:outputText value="Description :"  styleClass="text_label"/>
           <h:inputTextarea value="#{site.description}"
styleClass="textarea_desc" />
           <h:outputText value="Welcome Message :"  styleClass="text_label"/>
```

```
            <h:inputTextarea value="#{site.welcomeMessage}"
styleClass="textarea_desc" />
            <h:outputText value="Template : "  styleClass="text_label"/>
            <h:panelGrid columns="2">
              <h:outputText value="#{site.template}    " />
              <a4j:commandButton action="#{page.viewTemplates}"
value="Change" >
                <rich:toolTip styleClass="tooltip" value="Open page for managing
templates" />  </a4j:commandButton></h:panelGrid>
            <h:outputText value="Default 'No Result' message:"
styleClass="text_label" />
            <h:inputText value="#{site.noResultMessage}"
styleClass="textbox_form" />
            <f:facet name="footer">
              <a4j:commandButton action="#{site.updateSite}" value="Save"
onclick="if (!confirm('Update project settings?')) return false">
                <rich:toolTip styleClass="tooltip" value="Save changes to site
properties" />  </a4j:commandButton></f:facet>
          </h:panelGrid><br /><rich:separator /><br />
          <h:panelGrid columns="2" styleClass="panel_details"
            headerClass="view_subheader" footerClass="view_footer">
            <f:facet name="header">
              <h:outputText value="Database Settings" />            </f:facet>
            <h:outputText value="Database :"  styleClass="text_label"/>
            <h:inputText value="#{site.database}" styleClass="textbox_form"/>
            <h:outputText value="User: "  styleClass="text_label"/>
            <h:inputText value="#{site.user}" styleClass="textbox_form"/>
            <h:outputText value="Password :"  styleClass="text_label"/>
            <h:inputSecret value="#{site.password}" styleClass="textbox_form"/>
            <h:outputText value="Database prefix: "  styleClass="text_label"/>
            <h:inputText value="#{site.dbPrefix}" styleClass="textbox_form"/>
              <f:facet name="footer">
                <a4j:commandButton action="#{site.updateDb}" value="Save"
onclick="if (!confirm('#{site.askUpdateDB}')) return false" >
                <rich:toolTip styleClass="tooltip" value="Save changes to database
settings" /> </a4j:commandButton></f:facet></h:panelGrid>
</rich:panel>  </h:form></f:view></ui:define></ui:composition></html>
```

/admin/templates/default/default.css

```
/************** GENERAL ******************/
body {
    font-family: Tahoma, Arial, sans-serif;
    line-height: 1.3em;
    margin: 0;
    padding: 0;
    font-size: 10;
    color: #000000;
  background: #e5e6e6;
  background-image: url("bg.jpg");
  background-repeat: repeat-x;}
a:link, a:visited {
    text-decoration: underline;
    font-weight: normal;
  font-size: 12px;}
a:hover {
    text-decoration: none;
    font-weight: normal;
  font-size: 12px;}
/********** BODY **************/
table.main_body{
  width: 1000px;
  align: center;
  background-color: #F8F8F8;
  padding: 0px,0px,0px,0px;
  margin: 0px,0px,0px,0px;}
.content{
  width: 100%;
  align:center;
  text-align: center;
  padding-left: 30;
  font-size: 12px;}
.header {
  height: 70;
  width: 970px;
  padding-left: 30px;
  background: url("header2.jpg");
  font-weight:bolder;
  text-align: left;
  padding-top:20px;
  color: white;
  font-size: 40px;}
.footer{
  width: 100%;
  height: 50;
  padding-top:5px;;
  background:   #F62217;
  background: url("header2.jpg");
  text-align: center;
  font-size:10px;
  color: white;}
/**********MAIN MENU*************/
.menu_main{
  height: 20;
  width: 100%;
  border-width: 0px;
  color: white;
  font-size: 14px;
  text-align: left;
  background: black;
  padding-top: 3px;}
.rich-menu-item-folder{
  border-width: 0px;
  background: #E42217;}
.rich-menu-list-border{
  border-width: 0px;}
.rich-menu-list-bg{
  background: #E42217;}
.menu_dropdown{
  height: 20;
  color: white;
  font-size: 14px;
  text-align: left;
  background: black; }
.menu_item{
  color: black;
  font-size: 12px;
  text-align: left;
  background: #E42217;
  border-width: 0px;}
a:link.menu_main{
  color: white;
  font-size: 14px;
  font-weight: bold;
  padding-top: 3px;
  text-decoration: none;}
a:hover.menu_main{
  color: #F62217;}
/**********SUB MENU*************/
.menu_sub {
  background: black;
  height: 15;
  width: 100%;
  font-size: 12px;
  text-align: right;
  border-width: 0px;}
a:link.menu_sub{
  color: white;
  font-size: 12px;
  text-align: right;}
/********** HOMEPAGE ***********/
```

```
.home_header{
  font-size: 14px;
  height:20px;
  text-align: left;
  padding-top: 3px;}
.home_panel{
  text-align: justify;
  width:90%;}
/********** TEXT **************/
.title {
  font-weight: bolder;
  font-size: 16;
  padding-top: 15px;
  text-align: left;}
.text_label {
  font-weight: bold;
  font-size: 12px;}
.errorMessage {
  color: red;
  font-style: italic;
  font-size: 12px;
  text-align: center;}
.infoMessage{
  color: black;
  font-style: italic;
  font-size: 12px;
   text-align: center;}
   /************** USER ACCOUNT **************/
.table_login{
  width: 300px;}
.table_account{
  width: 400px;}
/************** VIEW **************/
.view_datatable{
  -moz-border-radius: 10px
  width: 90%;}
.view_header{
  font-size: 14px;
  height:20px;
  text-align: center;
  padding-top: 3px;}
.view_footer{
  text-align: center;}
.view_subheader{
  background: FF9999;
  font-weight: bold;
  font-size: 12px;
  height: 14px;
  text-align: center;}
.delete_columnHeader{
  background: FF9999;
  width: 20px;}
.panel_main{  width: 90%;}
.panel_details{
  text-align: left;
  width: 90%;
  padding-left: 20px;}
.panel_contents{
  text-align: center;
  width:100%;}
.panel_datatable{
  width:100%;}
.panel_footer{
  width: 100%;
  text-align: center;}
/************** OBJECT **************/
.delete_column{  width: 10px;}
.method_column{
  text-align: left;
   padding-left: 30px;}
   /************** RULESET **************/
.panel_rule{
  background-color: none;
  text-align: left;
  width: 100%;}
.column_rule{
  text-align: left;}
.panel_ifThen{
  width: 50px;
  font-size: 14px;
  text-align: right;
  vertical-align: top;
  padding-left: 20px;
  font-weight: bold;}
.panel_ruleAttr{
  text-align: right;}
.text_RuleId{
  font-size: 16px;
  text-align: left;
  padding-left: 10px;
  font-weight: bold;}
/********** FORMS **************/
#link_back a:link{
  font-weight: bold;
  font-size: 12;
  color: black;}
.textarea_desc{
  width: 300px;
  height: 50px;}
.textbox_name{
  width: 200px;}
.textbox_form{
  width: 300px;}
.textbox_number{
  width: 30px;}
.button_delete{
  font-size: 10;
  font-weight: bold;
  color: red;
  background:white;}
button:hover.button_delete{
  font-size: 10;
  background:red;
  color: black;
  font-weight: bold;}
.textarea_code{
  width: 405px;
  height: 100px;
  padding-left: 30px;
}
/********** RICH FACES ************/
.tooltip{
  color: black;
  border-color: red;
  background-color: #E77471;
  font-size: 9px
  white-space:nowrap;
  padding-top: 3px;
  padding-bottom: 3px;
  background-image: url("tooltip.jpg");}
```

/admin/user/add.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:a4j="http://richfaces.org/a4j"
      xmlns:rich="http://richfaces.org/rich">
<ui:composition template="../includes/template.xhtml">
  <ui:define name="title">Create New User</ui:define>
  <ui:define name="content">  <f:view><h:form>
```

```
      <rich:panel styleClass="table_account" headerClass="view_header">
      <f:facet name="header" >
         <h:outputText value="User Information" />  </f:facet>
      <h:panelGrid columns="2" styleClass="panel_details">
         <h:outputText value="Username: "  styleClass="text_label"/>
         <h:inputText value="#{user.username}" styleClass="textbox_name" />
<h:outputText value="Password: "  styleClass="text_label"/>
         <h:inputSecret value="#{user.password}" styleClass="textbox_name"
/> <h:outputText value="Name: "  styleClass="text_label"/>
         <h:inputText value="#{user.name}" styleClass="textbox_name" />
         <h:outputText value="Details: "  styleClass="text_label"/>
         <h:inputTextarea value="#{user.details}" styleClass="textarea_desc"/>
         <h:outputText value="User type: "  styleClass="text_label"/>
         <h:selectOneMenu value="#{user.type}">
            <f:selectItems value="#{user.selectUserTypes}" />
         </h:selectOneMenu>  </h:panelGrid>
      <h:panelGrid columns="1">
        <h:commandButton action="#{user.add}" value="Submit" />
      </h:panelGrid></rich:panel>
   <div id="link_back">
   <br/><h:commandLink value="BACK" action="#{page.viewUsers}"
styleClass="link_back"/>
  </div> </h:form></f:view> </ui:define></ui:composition></html>
```

/admin/user/index.jsp

```
<jsp:forward page="view.jsf" />
```

/admin/user/user.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
     xmlns:h="http://java.sun.com/jsf/html"
     xmlns:f="http://java.sun.com/jsf/core"
     xmlns:a4j="http://richfaces.org/a4j"
     xmlns:rich="http://richfaces.org/rich">
<ui:composition template="../includes/template.xhtml">
  <ui:define name="title">Create New User</ui:define>
  <ui:define name="content"> <f:view><h:form>
       <rich:panel styleClass="table_account">
       <f:facet name="header" >
          <h:outputText value="User Information" />          </f:facet>
       <h:panelGrid columns="2"  styleClass="panel_details">
          <h:outputText value="Username: "  styleClass="text_label"/>
          <h:inputText value="#{user.username}" styleClass="textbox_name"/>
          <h:outputText value="Name: "  styleClass="text_label"/>
          <h:inputText value="#{user.name}" styleClass="textbox_name"/>
          <h:outputText value="Details: "  styleClass="text_label"/>
          <h:inputTextarea value="#{user.details}" styleClass="textarea_desc"/>
          <h:outputText value="User type: "  styleClass="text_label"/>
          <h:selectOneMenu value="#{user.type}">
             <f:selectItems value="#{user.selectUserTypes}" />
          </h:selectOneMenu></h:panelGrid>
       <h:panelGrid columns="1">
         <h:commandButton action="#{user.update}" value="Save changes" />
       </h:panelGrid></rich:panel>
  <div id="link_back">
    <br/><h:commandLink value="BACK" action="#{page.viewUsers}"
styleClass="link_back"/>
  </div> </h:form></f:view></ui:define></ui:composition></html>
```

/admin/user/view.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
     xmlns:h="http://java.sun.com/jsf/html"
     xmlns:f="http://java.sun.com/jsf/core"
     xmlns:a4j="http://richfaces.org/a4j"
     xmlns:rich="http://richfaces.org/rich">
<ui:composition template="../includes/template.xhtml">
  <ui:define name="title">User Manager</ui:define>
  <ui:define name="content">  <f:view><h:form>
       <rich:dataTable value="#{table.users}" var="user"
        styleClass="view_datatable" headerClass="view_header"
        footerClass="view_footer">
         <f:facet name="header"> <h:outputText value="Users" /></f:facet>
         <rich:column headerClass="delete_columnHeader">
           <f:facet name="header">
             <h:commandButton action="#{table.deleteUsers}"
              onclick="if (!confirm('#{table.askDelUser}')) return false"
              value="delete" />
           </f:facet>
           <h:selectBooleanCheckbox value="#{user.removed}"
rendered="#{!user.userAdmin}"/></rich:column>
         <rich:column headerClass="view_subheader" id="username"
sortBy="#{user.username}">
           <f:facet name="header">
             <h:outputText value="Username" /></f:facet>
             <h:commandLink value="#{user.username}" action="#{link.viewUser}"
rendered="#{!user.userAdmin}">
               <f:param name="username" value="#{user.username}" />
             </h:commandLink>
             <h:outputText value="#{user.username}"
rendered="#{user.userAdmin}" /></rich:column>
         <rich:column headerClass="view_subheader" id="name"
sortBy="#{user.name}">
           <f:facet name="header">
             <h:outputText value="Name" /></f:facet>
             <h:outputText value="#{user.name}" /></rich:column>
         <rich:column headerClass="view_subheader" id="details"
sortBy="#{user.details}">
           <f:facet name="header">
             <h:outputText value="Details" /></f:facet>
             <h:outputText value="#{user.details}" /></rich:column>
         <rich:column headerClass="view_subheader" id="type"
sortBy="#{user.stringType}">
           <f:facet name="header">
             <h:outputText value="User Type" /></f:facet>
             <h:outputText value="#{user.stringType}" /></rich:column>
         <rich:column headerClass="view_subheader" id="status"
sortBy="#{user.stringStatus}">
           <f:facet name="header">
             <h:outputText value="Status" /></f:facet>
               <h:selectBooleanCheckbox value="#{user.status}"
rendered="#{!user.userAdmin}"
                 onclick="if (!confirm('#{user.askStatus}')) return false" >
                <a4j:support event="onchange" action="#{user.activate}"
reRender="status, hMessage" /></h:selectBooleanCheckbox> </rich:column>
           <f:facet name="footer">
         <h:commandButton action="#{page.addUser}" value="Create new user"
/></f:facet> </rich:dataTable> </h:form></f:view></ui:define>
</ui:composition></html>
```

/includes/footer.xhtml

```
<ui:composition xmlns:f="http://java.sun.com/jsf/core"
              xmlns:h="http://java.sun.com/jsf/html"
              xmlns:ui="http://java.sun.com/jsf/facelets">
<f:view> <h:form>
<b><h:outputText value="#{page.siteName}" /></b>
<h:outputText value=" - #{page.siteDesc}" /><b>
<br />All Rights Reserved<br />Powered by GEEBBO</b>
</h:form></f:view></ui:composition>
```

/includes/header.xhtml

```
<ui:composition xmlns:f="http://java.sun.com/jsf/core"
              xmlns:h="http://java.sun.com/jsf/html"
              xmlns:ui="http://java.sun.com/jsf/facelets">
<f:view> <h:form></h:form></f:view></ui:composition>
```

/includes/menu.xhtml

```
<ui:composition xmlns:f="http://java.sun.com/jsf/core"
              xmlns:h="http://java.sun.com/jsf/html"
              xmlns:ui="http://java.sun.com/jsf/facelets"
              xmlns:rich="http://richfaces.org/rich">
<f:view><h:form>
```

```
<h:panelGrid styleClass="header">
  <h:outputText value="#{page.siteName}"/>
</h:panelGrid>
<rich:toolBar itemSeparator="line" styleClass="main_menu">
  <h:commandLink action="home" value="Home" styleClass="menu"/>
  <h:commandLink action="rules" value="Rulesets" styleClass="menu"/>
</rich:toolBar></h:form></f:view></ui:composition>
```

/includes/template.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  xmlns:f="http://java.sun.com/jsf/core"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:a4j="http://richfaces.org/a4j"
  xmlns:rich="http://richfaces.org/rich">
<head><title><ui:insert name="title" /></title>
    <link rel="stylesheet" type="text/css"
href="/geebbo/templates/#{page.frontTemplate}" />
</head><body><h:panelGrid styleClass="panel_main">
  <ui:include src="menu.xhtml"/>
  <h:panelGrid styleClass="content">
    <h:messages errorClass="errorMessage" infoClass="infoMessage"
      layout="table" globalOnly="true" showDetail="false"
      showSummary="true" /> <ui:insert name="content" /></h:panelGrid>
  <h:panelGrid styleClass="footer">
    <ui:include src="footer.xhtml"/>
  </h:panelGrid></h:panelGrid></body></html>
```

/rules/form.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:a4j="http://richfaces.org/a4j"
      xmlns:rich="http://richfaces.org/rich">
<ui:composition template="../includes/template.xhtml">
  <ui:define name="title">View Rules</ui:define>
  <ui:define name="content"> <f:view><h:form>
   <rich:panel styleClass="panel_content" headerClass="panel_header">
     <f:facet name="header">
      <h:outputText value="Ruleset: #{ruleset.name}" />    </f:facet>
          <rich:dataTable value="#{ruleset.arrlForms}" var="item"
                headerClass="tableHeader" styleClass="tableClass"
                footerClass="tableFooter">
                <f:facet name="header">
                   <h:outputText value="Please answer the following questions:"
/> </f:facet>
                <h:column>
                   <f:facet name="header">
                      <h:outputText value="Question" /></f:facet>
                   <h:outputText value="#{item.question}" /></h:column>
                <h:column><!--  set input types  -->
                   <h:inputText rendered="#{item.inputText}" value="#{item.value}"
/>
                   <h:selectManyCheckbox rendered="#{item.checkBox}"
value="#{item.values}">
                      <f:selectItems value="#{item.selectItems}" />
                   </h:selectManyCheckbox>
                   <h:selectOneMenu rendered="#{item.menu}"
value="#{item.value}">
                      <f:selectItems value="#{item.selectItems}" />
                   </h:selectOneMenu>
                   <h:selectOneRadio rendered="#{item.radioButton}"
value="#{item.value}">
                      <f:selectItems value="#{item.selectItems}" />
                   </h:selectOneRadio></h:column>
                 <f:facet name="footer">
                   <a4j:commandButton action="#{ruleset.runEngine}"
value="Submit" styleClass="buttonClass">
              <rich:toolTip value="Submit data and view results"
styleClass="tooltipClass"/></a4j:commandButton></f:facet></rich:dataTable>
        <h:panelGrid styleClass="tableFooter">
            <a4j:commandButton value="Back" action="rules"
styleClass="buttonClass" >
            <rich:toolTip value="Go back to list of rulesets"
styleClass="tooltipClass"/></a4j:commandButton></h:panelGrid> </rich:panel>
</h:form></f:view> </ui:define></ui:composition></html>
```

/rules/result.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
       xmlns:h="http://java.sun.com/jsf/html"
       xmlns:f="http://java.sun.com/jsf/core"
      xmlns:a4j="http://richfaces.org/a4j"
      xmlns:rich="http://richfaces.org/rich">
<ui:composition template="../includes/template.xhtml">
   <ui:define name="title">View Rules</ui:define>
   <ui:define name="content"><f:view><h:form>
   <rich:panel styleClass="panel_content" headerClass="panel_header">
    <f:facet name="header">
     <h:outputText value="Result for Ruleset: #{ruleset.name}" />   </f:facet>
    <h:outputText rendered="#{!ruleset.decided}"
     value="#{ruleset.stringNoDecision }" />
    <rich:dataTable value="#{ruleset.result.decisions}" var="dec"
     headerClass="tableHeader" styleClass="tableClass"
     footerClass="tableFooter" rendered="#{!ruleset.decisionEmpty}">
     <rich:column>
      <f:facet name="header">
       <h:outputText value="Decision(s):"  />      </f:facet>
      <h:outputText value="#{dec.decision}" /> </rich:column>
     <rich:column>
      <f:facet name="header">
       <h:outputText value="Reason(s):"  />     </f:facet>
      <ul>  <ui:repeat value="#{dec.conditions}" var="con">
         <li> <h:outputText value="#{con}" /> </li>
        </ui:repeat> </ul> </rich:column> </rich:dataTable>
    <rich:dataTable value="#{ruleset.result.output}" var="out"
     headerClass="tableHeader" styleClass="tableClass"
     footerClass="tableFooter" rendered="#{!ruleset.outputEmpty}">
     <f:facet name="header">
      <h:outputText value="Output(s):"  /></f:facet>
     <rich:column>
      <h:outputText value="#{out.object}'s " />
      <h:outputText value="#{out.attribute} : " />
      <h:outputText value="#{out.value}" /></rich:column>
    </rich:dataTable>
        <h:panelGrid styleClass="tableFooter">
      <a4j:commandButton value="Back to input form" action="form"
styleClass="buttonClass">
       <rich:toolTip value="Go back to page for answering questions"
styleClass="tooltipClass"/> </a4j:commandButton></h:panelGrid>
</rich:panel></h:form></f:view></ui:define></ui:composition></html>
```

/rules/view.jsp

```
<html xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:a4j="http://richfaces.org/a4j"
      xmlns:rich="http://richfaces.org/rich">
<ui:composition template="../includes/template.xhtml">
  <ui:define name="title">View Rules</ui:define>
  <ui:define name="content">  <f:view><h:form>
   <rich:panel styleClass="panel_content" headerClass="panel_header">
   <f:facet name="header">
    <h:outputText value="Rulesets" />    </f:facet>
        <rich:dataTable value="#{table.publishedRulesets}" var="rs"
           headerClass="tableHeader" styleClass="tableClass"
           footerClass="tableFooter">
           <rich:column sortBy="#{rs.name}">
            <f:facet name="header">
              <h:outputText value="Ruleset">
```

```
            <rich:toolTip value="Sort rulesets by name"
styleClass="tooltipClass"/> </h:outputText></f:facet>
                  <h:commandLink value="#{rs.name}"
action="#{link.viewInputForm}">
                     <f:param name="ruleset" value="#{rs.id}" />
                  </h:commandLink></rich:column>
        <rich:column sortBy="#{rs.stringRulegroup}">
          <f:facet name="header">
            <h:outputText value="Rule Group" >
              <rich:toolTip value="Sort rulesets by rule group"
styleClass="tooltipClass"/></h:outputText> </f:facet>
          <h:outputText value="#{rs.stringRulegroup}" />
        </rich:column>
              <rich:column sortBy="#{rs.description}">
                <f:facet name="header">
              <h:outputText value="Description" >
              <rich:toolTip value="Sort rulesets by description"
styleClass="tooltipClass"/></h:outputText> </f:facet>
            <h:outputText value="#{rs.description}" />
               </rich:column></rich:dataTable></rich:pane  </h:form></f:view>
  </ui:define></ui:composition></html>
```

/templates/default/default.css

```
body {
    font-family: Tahoma, Arial, sans-serif;
    line-height: 1.3em;
    margin: 0;
    padding: 0;
    font-size: 12px;
  background: #AFC7C7;}
a:link {  color:  #5E7D7E;}
a:hover {  color:  #5E7D7E;}
/*** Template specific elements ***/
.header {
  width: 100%;
  height: 60;
  background: #3B9C9C;
  color: white;
  font-size: 30;
  font-weight:bolder;
  text-align: left;
  padding-top:15;
  text-align: center;}
.footer{
  width: 100%;
  height: 35px;
  padding-top: 5px;
  background: #307D7E;
  text-align: center;
  font-size: 10px;
  color: white;}
.main_menu{
  width: 100%;
  height: 30;
  background: #92C7C7;
  border-width: 0px;}
a:link.menu{
  font-size: 16px;
  font-style: none;
  color: white;}
   /******** TEXT **********/
.errorMessage{
  color: red;
  font-size: 14px;
  font-style: italic;}
.infoMessge {
  color: black;
  font-size: 14px;
  font-style: italic;}
/******** COMPONENTS *********/
.panel_main{
  width: 100%;
  align: center;
  border-width: 0px;
  padding: 0px;
  margin: 0px;
  background: #e5e6e6;}
.content{
  width: 100%;
  color:black;
  padding-left: 20px;
  padding-right: 20px;
  padding-bottom: 20px;
  min-height: 200;
  text-align: center;}
.panel_content{
  width:90%;
  border-width: 0px;
  font-size: 14px;
  text-align: justify;
  padding-bottom: 10px;}
.panel_header{
  background: #566D7E;
  font-size: 16px;
  font-weight: bold;
  border-width: 0px;}
.tableClass{
  border-width: 0px;
  width: 100%;}
.tableHeader{  background: #566D7E;}
.tableFooter{
  background: #AFC7C7;
  text-align:center;
  width: 100%;}
.tooltipClass{
  font-size: 9px;
  border-color: cyan;
  color: black;
  background: #e5e6e6;}
.buttonClass {
  background: #566D7E;
  color: white;}
```

/WEB-INF/faces-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-facesconfig_1_2.xsd"
    version="1.2">
<application>
   <view-handler>
      com.sun.facelets.FaceletViewHandler   </view-handler>
</application>
  <managed-bean>
    <managed-bean-name>sessionbean</managed-bean-name>
    <managed-bean-class>beans.SessionBean</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
  </managed-bean>  <managed-bean>
    <managed-bean-name>page</managed-bean-name>
    <managed-bean-class>beans.PageManager</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
  </managed-bean>  <managed-bean>
    <managed-bean-name>site</managed-bean-name>
    <managed-bean-class>beans.SiteConfiguration</managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
  </managed-bean>  <managed-bean>
    <managed-bean-name>template</managed-bean-name>
    <managed-bean-class>beans.Template</managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
```

```
</managed-bean>    <managed-bean>
  <managed-bean-name>user</managed-bean-name>
  <managed-bean-class>beans.User</managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>    <managed-bean>
     <managed-bean-name>table</managed-bean-name>
     <managed-bean-class>beans.DataTable</managed-bean-class>
     <managed-bean-scope>request</managed-bean-scope>
  </managed-bean>  <managed-bean>
     <managed-bean-name>object</managed-bean-name>
     <managed-bean-class>beans.ObjectBean</managed-bean-class>
     <managed-bean-scope>session</managed-bean-scope>
  </managed-bean>  <managed-bean>
     <managed-bean-name>attribute</managed-bean-name>
     <managed-bean-class>beans.Attribute</managed-bean-class>
     <managed-bean-scope>request</managed-bean-scope>
  </managed-bean>  <managed-bean>
  <managed-bean-name>function</managed-bean-name>
  <managed-bean-class>beans.Function</managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
</managed-bean>    <managed-bean>
  <managed-bean-name>parameter</managed-bean-name>
  <managed-bean-class>beans.Parameter</managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
</managed-bean> <managed-bean>
  <managed-bean-name>rulegroup</managed-bean-name>
  <managed-bean-class>beans.Rulegroup</managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
</managed-bean>    <managed-bean>
  <managed-bean-name>ruleset</managed-bean-name>
  <managed-bean-class>beans.Ruleset</managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>    <managed-bean>
  <managed-bean-name>rule</managed-bean-name>
  <managed-bean-class>beans.Rule</managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
</managed-bean>    <managed-bean>
  <managed-bean-name>condition</managed-bean-name>
  <managed-bean-class>beans.Condition</managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
</managed-bean><managed-bean>
  <managed-bean-name>consequence</managed-bean-name>
  <managed-bean-class>beans.Consequence</managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
</managed-bean>  <managed-bean>
  <managed-bean-name>result</managed-bean-name>
  <managed-bean-class>beans.Result</managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
</managed-bean><managed-bean>
  <managed-bean-name>decision</managed-bean-name>
  <managed-bean-class>beans.Decision</managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
</managed-bean>  <managed-bean>
  <managed-bean-name>input</managed-bean-name>
  <managed-bean-class>beans.InputForm</managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
</managed-bean> <managed-bean>
  <managed-bean-name>link</managed-bean-name>
  <managed-bean-class>beans.CommandLink</managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
  <managed-property>
    <property-name>object</property-name>
    <value>#{param.object}</value>
  </managed-property>    <managed-property>
    <property-name>ruleset</property-name>
    <value>#{param.ruleset}</value>
  </managed-property>    <managed-property>
    <property-name>username</property-name>
    <value>#{param.username}</value>
  </managed-property>
</managed-bean>
  <navigation-rule>
     <from-view-id>*</from-view-id>
     <navigation-case>
        <from-outcome>home</from-outcome>
        <to-view-id>/home.jsp</to-view-id>
     </navigation-case>
     <navigation-case>
        <from-outcome>rules</from-outcome>
        <to-view-id>/rules/view.jsp</to-view-id>
     </navigation-case>
  </navigation-rule>
  <navigation-rule>
     <from-view-id>/admin/*</from-view-id>
     <navigation-case>
        <from-outcome>home</from-outcome>
        <to-view-id>/admin/home.jsp</to-view-id>
     </navigation-case>
  <navigation-case>
    <from-outcome>objects</from-outcome>
    <to-view-id>/admin/object/view.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>addobjects</from-outcome>
    <to-view-id>/admin/object/add.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>viewobjects</from-outcome>
    <to-view-id>/admin/object/object.jsp</to-view-id>
  </navigation-case>
     <navigation-case>
        <from-outcome>rules</from-outcome>
        <to-view-id>/admin/rule/view.jsp</to-view-id>
     </navigation-case>
  <navigation-case>
    <from-outcome>addrules</from-outcome>
    <to-view-id>/admin/rule/add.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>viewrules</from-outcome>
    <to-view-id>/admin/rule/rule.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>rulegroups</from-outcome>
    <to-view-id>/admin/rule/group.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>formrules</from-outcome>
    <to-view-id>/admin/rule/form.jsp</to-view-id>
  </navigation-case>
     <navigation-case>
        <from-outcome>users</from-outcome>
        <to-view-id>/admin/user/view.jsp</to-view-id>
     </navigation-case>
  <navigation-case>
    <from-outcome>viewusers</from-outcome>
    <to-view-id>/admin/user/user.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>addusers</from-outcome>
    <to-view-id>/admin/user/add.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>settings</from-outcome>
    <to-view-id>/admin/setting/view.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>templates</from-outcome>
    <to-view-id>/admin/setting/templates.jsp</to-view-id>
  </navigation-case>
```

```
    <navigation-case>
      <from-outcome>account</from-outcome>
      <to-view-id>/admin/account/account.jsp</to-view-id>
    </navigation-case>
  <navigation-case>
   <from-outcome>login</from-outcome>
   <to-view-id>/admin/login.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
   <from-outcome>pwdforgot</from-outcome>
   <to-view-id>/admin/forgot.jsp</to-view-id>
  </navigation-case>
  </navigation-rule>
  <navigation-rule>
    <from-view-id>/admin/user/*</from-view-id>
  <navigation-case>
   <from-outcome>user</from-outcome>
   <to-view-id>/admin/user/user.jsp</to-view-id>
  </navigation-case>
  </navigation-rule>
 <navigation-rule>
  <from-view-id>/admin/account/*</from-view-id>
  <navigation-case>
   <from-outcome>changepwd</from-outcome>
   <to-view-id>/admin/account/changePwd.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
   <from-outcome>changeqa</from-outcome>
   <to-view-id>/admin/account/changeQA.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
   <from-outcome>changed</from-outcome>
   <to-view-id>/admin/account/account.jsp</to-view-id>
  </navigation-case>
 </navigation-rule>
  <navigation-rule>
    <from-view-id>/admin/object/*</from-view-id>
    <navigation-case>
      <from-outcome>object</from-outcome>
      <to-view-id>/admin/object/object.jsp</to-view-id>
    </navigation-case>
  </navigation-rule>
   <navigation-rule>
  <from-view-id>/admin/rule/*</from-view-id>
  <navigation-case>
   <from-outcome>rule</from-outcome>
   <to-view-id>/admin/rule/rule.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
   <from-outcome>form</from-outcome>
   <to-view-id>/admin/rule/form.jsp</to-view-id>
  </navigation-case>
 </navigation-rule>
  <navigation-rule>
  <from-view-id>/rules/*</from-view-id>
  <navigation-case>
   <from-outcome>form</from-outcome>
   <to-view-id>/rules/form.jsp</to-view-id>
  </navigation-case>
  <navigation-case>
   <from-outcome>result</from-outcome>
   <to-view-id>/rules/result.jsp</to-view-id>
  </navigation-case>
 </navigation-rule>
</faces-config>
```

/WEB-INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
 id="WebApp_ID" version="2.5">
 <display-name>geebbo</display-name>
 <welcome-file-list>
  <welcome-file>index.html</welcome-file>
  <welcome-file>index.htm</welcome-file>
  <welcome-file>index.jsp</welcome-file>
  <welcome-file>default.html</welcome-file>
  <welcome-file>default.htm</welcome-file>
  <welcome-file>default.jsp</welcome-file>
 </welcome-file-list>
<filter>
 <filter-name>PageFilter</filter-name>
 <filter-class>general.PageFilter</filter-class>
</filter>
<filter-mapping>
 <filter-name>PageFilter</filter-name>
 <url-pattern>/admin/*</url-pattern>
</filter-mapping>
<listener>
  <listener-class>com.sun.faces.config.ConfigureListener</listener-class>
</listener>
<context-param>
 <param-name>facelets.REFRESH_PERIOD</param-name>
 <param-value>2</param-value>
</context-param>
<context-param>
 <param-name>facelets.DEVELOPMENT</param-name>
 <param-value>true</param-value>
</context-param>
<!-- RICH FACES -->
<context-param>
  <param-name>org.richfaces.SKIN</param-name>
  <param-value>ruby</param-value>
</context-param>
<context-param>
    <param-name>org.richfaces.CONTROL_SKINNING</param-name>
    <param-value>enable</param-value>
</context-param>
<filter>
  <display-name>RichFaces Filter</display-name>
  <filter-name>richfaces</filter-name>
  <filter-class>org.ajax4jsf.Filter</filter-class>
</filter>
<filter-mapping>
  <filter-name>richfaces</filter-name>
  <servlet-name>Faces Servlet</servlet-name>
  <dispatcher>REQUEST</dispatcher>
  <dispatcher>FORWARD</dispatcher>
  <dispatcher>INCLUDE</dispatcher>
</filter-mapping>
 <!-- Faces Servlet -->
 <servlet>
  <servlet-name>Faces Servlet</servlet-name>
  <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
 </servlet>
 <!-- Faces Servlet Mapping -->
 <servlet-mapping>
  <servlet-name>Faces Servlet</servlet-name>
  <url-pattern>*.jsf</url-pattern>
 </servlet-mapping>
</web-app>
```

/WEB-INF/classes/beans/Attribute.java

```
package beans;
```

```
import general.Globals;import java.sql.SQLException;import
database.DbManager;
public class Attribute {
   // database columns
   private int id;
   private int objId;
   private int datatype;
   private String variable = "";
   private String name = "";
   private String description = "";
   // helper
   private boolean removed = false;
   private boolean added = false;
   /*  * CONSTRUCTORS   */
   public Attribute() {   }
   public Attribute(int objId, boolean added){
      this.objId = objId;
      this.added = added;   }
   /*  * METHODS   */
   public void save(int objId) throws SQLException {
      this.objId = objId;
      DbManager.getInstance().addAttribute(this); }
   public void update() throws SQLException {
      DbManager.getInstance().updateAttribute(this);   }
   public void updateVariable(){
      variable = name.trim().replace(" ", "_"); }
   public void removeThis() {      removed = true;  }
   public boolean isRemoved() {    return removed;  }
   public void setAdded(boolean added){      this.added = added;   }
   public boolean isAdded(){      return added;  }
   public String getStringDatatype() throws SQLException{
      return Globals.getDataType(datatype); }
   /*  * GETTERS & SETTERS   */
   public int getId() {       return id; }
   public void setId(int id) {     this.id = id;  }
   public int getObjId() {      return objId;}
   public void setObjId(int objId) {      this.objId = objId;}
   public int getDatatype() {    return datatype;  }
   public void setDatatype(int datatype) {      this.datatype = datatype;}
   public String getVariable() {       return variable;   }
   public void setVariable(String variable) {       this.variable = variable;  }
   public String getName() {       return name;   }
   public void setName(String name) {       this.name = name;  }
   public String getDescription() {       return description;   }
   public void setDescription(String description) {     this.description =
description; }}
```

/WEB-INF/classes/beans/CommandLink.java

```
package beans;
import java.io.IOException;import java.sql.SQLException;
import javax.faces.context.FacesContext;import database.DbManager;
public class CommandLink {
   private String object;
   private String ruleset;
   private String username;
   /*  * CONSTRUCTOR */
   public CommandLink() { }
   /*  * METHODS   */
   public String viewObject() throws NumberFormatException, SQLException,
IOException {
      ObjectBean ob =
DbManager.getInstance().getObject(Integer.parseInt(object));
      ob.init();
      setSessionObj("object", ob);
      return "view"+PageManager.OBJECTS;   }
   public String viewRule() throws NumberFormatException, SQLException,
IOException {
      Ruleset r =
DbManager.getInstance().getRuleset(Integer.parseInt(ruleset));
      r.init();
      setSessionObj("ruleset", r);
      return "view"+PageManager.RULES;}
   public String viewForm() throws SQLException, NumberFormatException,
IOException {
      Ruleset r =
DbManager.getInstance().getRuleset(Integer.parseInt(ruleset));
      r.setArrlForms(DbManager.getInstance().getForms(r.getId(), true));
      setSessionObj("ruleset", r);
      return "form"+PageManager.RULES;}
   /*  * FOR FRONT-END   */
   public String viewInputForm() throws NumberFormatException,
SQLException, IOException{
      Ruleset r =
DbManager.getInstance().getRuleset(Integer.parseInt(ruleset));
      r.setArrlForms(DbManager.getInstance().getForms(r.getId(), false));
      setSessionObj("ruleset", r);
      return PageManager.FORM;  }
   public String viewUser() throws SQLException, IOException{
      User user = DbManager.getInstance().getUser(username);
      setSessionObj("user", user);
      return "view"+PageManager.USERS;   }
   /*  * FACES MAP    */
   private void setSessionObj(String key, Object o) {
   FacesContext.getCurrentInstance().getExternalContext().getSessionMap().p
ut(key, o);   }
   /*  * GETTERS & SETTERS   */
   public String getObject() {      return object;  }
   public void setObject(String object) {     this.object = object;}
   public String getRuleset() {     return ruleset; }
   public void setRuleset(String rule) {       this.ruleset = rule;   }
   public String getUsername() {    return username;}
   public void setUsername(String username) { this.username = username;}}
```

/WEB-INF/classes/beans/Condition.java

```
package beans;
import general.Globals;import java.sql.SQLException;import
java.util.ArrayList;import javax.faces.model.SelectItem;import
database.DbManager;
public class Condition {
   //database columns
   private int id;
   private int ruleId; //rule primary key
   private int objId;
   private int attrId;
   private String function;
   private int optrId;
   private String value="";
   private int type = 1; //1=obj.attr to value, 2=obj.attr to function, 3=function to
function, 4=function to value
   private int bool = -1;    //0 for OR, 1 for AND
   private int seqNo; //order of condition
   //helpers
   private boolean removed = false;
   private boolean added = true;
   private int to;
   private int from;
   /*  * CONSTRUCTORS   */
   public Condition(){  }
      public Condition(int ruleId, int order){
      this.ruleId = ruleId;
      this.seqNo = order; }
   public Condition(int bool, int ruleId, int order, int startObj){
      this.ruleId = ruleId;
      this.bool = bool;
      this.seqNo = order;
      added = true;
      objId = startObj;  }
   /*  * METHODS   */
   public void save(int ruleId) throws SQLException{
      this.ruleId = ruleId;
```

```
    if(!isShowFuncLeft())
       function = "";   //no need to save function name
    if(isShowToFrom())
       value = from+","+to;
    DbManager.getInstance().addCondition(this);   }
  public void update() throws SQLException{
    if(!isShowFuncLeft())
       function = "";   //no need to save function name
    if(isShowToFrom())
       value = from+","+to;
    DbManager.getInstance().updateCondition(this);  }
  /*  * BOOLEAN CHECKERS   */
  public void removeThis(){       removed = true;  }
  public boolean isRemoved(){     return removed;  }
  public boolean isAdded(){       return added;  }
  public void setAdded(boolean added){this.added = added;  }
  public boolean isShowObjAttr(){    if(type<=2)   return true;
    else return false; }
  public boolean isShowFuncLeft(){  if(type>=3)   return true;
    else return false; }
  public boolean isShowFuncRight(){   if(type==2 || type==3)     return true;
    else return false; }
  public boolean isShowInputText(){
    if( (type==1 || type==4) && !isShowToFrom())   return true;
    else return false; }
  public boolean isShowToFrom(){
    return (optrId>=10 && (type==1 || type==4)); }
  /*  * GETTERS    */
  public String getStringObject() throws SQLException{
    return Globals.getObjectName(objId);   }
  public String getStringAttribute() throws SQLException{
    if(attrId==0){    return " "; }
    else{    return Globals.getAttributeName(attrId);    }  }
  public String getStringOperator() throws SQLException{
    return Globals.getOperator(optrId);    }
  public String getStringBool(){     if(bool==1)      return "and";
    else if(bool==0)   return "or";      else return "";  }
  public ArrayList<SelectItem> getAttributes() throws SQLException{
    if(objId>0)       return Globals.getSelectAttributes(objId);
    else     return new ArrayList<SelectItem>();  }
  /*  * GETTERS & SETTERS   */
  public int getRuleId() {    return ruleId;   }
  public int getId() {       return id; }
  public void setId(int id) {     this.id = id;  }
  public void setRuleId(int ruleId) {       this.ruleId = ruleId;  }
  public int getObjId() {      return objId;}
  public void setObjId(int objId) {      this.objId = objId;}
  public int getAttrId() {      return attrId;}
  public void setAttrId(int attrId) {      this.attrId = attrId;   }
  public String getFunction() {       return function;   }
  public void setFunction(String function) {       this.function = function;  }
  public int getOptrId() {     return optrId;   }
  public void setOptrId(int optrId) {       this.optrId = optrId; }
  public String getValue() {       return value;   }
  public void setValue(String value) {       this.value = value;  }
  public int getBool() {       return bool; }
  public void setBool(int bool) {     this.bool = bool;  }
  public int getType() {       return type; }
  public void setType(int type){     this.type = type;  }
  public int getSeqNo() {       return seqNo;  }
  public void setSeqNo(int seqNo) {      this.seqNo = seqNo;   }
  public int getTo() {       return to; }
  public void setTo(int to) {    this.to = to;  }
  public int getFrom() {      return from; }
  public void setFrom(int from) {       this.from = from; }}
```

/WEB-INF/classes/beans/Consequence.java

```
package beans;
import general.Globals;import general.Messages;import
java.sql.SQLException;import java.util.ArrayList;import
javax.faces.model.SelectItem;import database.DbManager;
public class Consequence {
   //database columns
   private int id;
   private int ruleId; //from db
   private int type = 1;
   private int seqNo;
   private String value=""; //operand or next rule
   private int objId;
   private int attrId;
   private String function;       //for function call
   //helpers
   private boolean added = false;
   private boolean showMethod = false;
   private String temp;
   private int valOrFunc;
   /*  * CONSTRUCTORS   */
   public Consequence(){}
   public Consequence(int ruleId, boolean added, int seqNo, int startObj){
      this.ruleId= ruleId;
      this.added = added;
      this.seqNo = seqNo;
      objId = startObj;  }
   /*  * METHODS   */
   public void save(int ruleId) throws SQLException{
      this.ruleId = ruleId;
      if(!showMethod)
         function = "";   //no need to save method id
      DbManager.getInstance().addConsequence(this);    }
   public void update() throws SQLException{
      if(!showMethod)
         function = "";
      DbManager.getInstance().updateConsequence(this);  }
   public void onChange(){     if(valOrFunc==0)        showMethod = false;
      else showMethod = true;}
   public String getBtnLabel(){
      if(!showMethod)
          return Messages.getMessage("conseq_showMethod");
      else{
         if(type==1)
            return "Remove function call";
         else return Messages.getMessage("conseq_hideMethod"); }  }
   /*  * CONSEQUENCE TYPE CHECKERS */
   public boolean isShowInputText() throws SQLException{
      temp = Globals.getConseq(type);
      return temp.equals("decision") || (temp.equals("modify")
&& !showMethod);  }
   public boolean isShowObjectAttr() throws SQLException{
      temp = Globals.getConseq(type);
      return (temp.equals("output value") || temp.equals("modify"));  }
   public boolean isShowMethods(){  return showMethod;   }
   public boolean isModify() throws SQLException{
      return Globals.getConseq(type).equals("modify"); }
   public boolean isShowRules() throws SQLException{
      return Globals.getConseq(type).equals("go to");   }
   public boolean isDecision() throws SQLException{
      return (Globals.getConseq(type).equals("decision"));   }
   /*  * SELECT ITEMS   */
   public ArrayList<SelectItem> getAttributes() throws SQLException{
      return Globals.getSelectAttributes(objId);  }
   /*  * STRING GETTERS */
   public String getStringConsequence() throws SQLException{
      return Globals.getConseq(type);}
   public String getStringAttribute() throws SQLException{
      return Globals.getAttributeName(attrId);    }
   public String getStringObject() throws SQLException{
      return Globals.getObjectName(objId);   }
   /*  * GETTERS & SETTERS   */
```

```
    public int getId() {       return id; }
    public void setId(int id) {     this.id = id;  }
    public int getRuleId() {    return ruleId;   }
    public void setRuleId(int ruleId) {       this.ruleId = ruleId;  }
    public int getType() {      return type; }
    public void setType(int type) {    this.type = type;  }
    public int getSeqNo() {       return seqNo;  }
    public void setSeqNo(int seqNo) {     this.seqNo = seqNo;   }
    public String getValue() {       return value;   }
    public void setValue(String value) {       this.value = value;  }
    public int getObjId() {      return objId;}
    public void setObjId(int objId) {      this.objId = objId;}
    public int getAttrId() {      return attrId;}
    public void setAttrId(int attrId) {      this.attrId = attrId;   }
    public String getFunction() {       return function;   }
    public void setFunction(String function) {   this.function = function;
      if(!function.isEmpty())     showMethod = true;   }
    public boolean isAdded(){      return added;  }
    public int getValOrFunc() {     return valOrFunc;   }
    public void setValOrFunc(int valOrFunc) {     this.valOrFunc = valOrFunc;}}
```

/WEB-INF/classes/beans/DataTable.java

```
package beans;
import general.FileManager;import general.Globals;import
general.Messages;import java.io.IOException;import java.sql.SQLException;
import java.util.ArrayList;import database.DbManager;import
drools.main.REManager;
public class DataTable {
    public DataTable(){}
    /*  * ================== USERS */
    private ArrayList<User> users;
    public ArrayList<User> getUsers() {
        try {
            users = DbManager.getInstance().getUsers();
        } catch (SQLException e) {
            Messages.err_gets("users");
            e.printStackTrace();
            users = new ArrayList<User>();     }
        return users;   }
    public String deleteUsers(){
        //get selected users
        String usernames = "";
        int ctr = 0;
        for(User u: users){
            if(u.isRemoved()){
                usernames += u.getUsername() + ",";
                ctr++;  }  }
        if (!usernames.isEmpty()) {
            try {
                DbManager.getInstance().deleteUsers(usernames.split(","));
                Messages.suc_delete(ctr+" user(s)");
            } catch (SQLException e) {
                Messages.err_gets("users");
                e.printStackTrace();
            }      }
        return "";  }
    public String getAskDelUser(){
        return Messages.getConfirmDel("users");  }
    /*  * ================== OBJECTS */
    private ArrayList<ObjectBean> objects;
    public ArrayList<ObjectBean> getObjects() {
        try {
            objects = DbManager.getInstance().getObjects();
        } catch (SQLException e) {
            Messages.err_gets("objects");
            e.printStackTrace();
            objects = new ArrayList<ObjectBean>();       }
        return objects; }
        public String deleteObjects() {
        // get selected objects
        String ids = "";
        String names = "";
        int ctr=0;
        for(ObjectBean o: objects){
            if(o.isRemoved()){
                ids += o.getId() + ",";
                names += o.getName() + ",";
                ctr++;          }      }
        if(!ids.isEmpty()){//assumed "names" also not empty
            try {
            //delete files from db
            DbManager.getInstance().deleteObjects(ids.split(","));
            //delete actual files
            for(String name: names.split(","))
                FileManager.deleteObject(name);
            Messages.suc_delete(ctr+" object(s)");
            REManager.updatePublishedRuleset();
            Globals.initHashObjAttrs();
            //update list of rules in case there are automatically delete attributes
            } catch (SQLException e){
                Messages.err_delete(" Database object(s)");
                e.printStackTrace();
            } catch (IOException e) {
                Messages.err_delete(" File(s)");
                e.printStackTrace();            }      }
        return "";  }
    public String getAskDelObj(){
        return "Affected rulesets will be automatically deleted. "+
Messages.getConfirmDel("objects");  }
    /*  * ================== PUBLISHED RULESETS */
    public ArrayList<Ruleset> getPublishedRulesets(){
        ArrayList<Ruleset> rulesets;
        try {
            rulesets = REManager.getPublishedRulesets();
        } catch (SQLException e) {
            Messages.err_gets("published ruleset.\n"+e.getMessage());
            e.printStackTrace();
            rulesets = new ArrayList<Ruleset>();     }
        return rulesets;   }
    //============= ALL RULESETS
    private ArrayList<Ruleset> rulesets;
    public ArrayList<Ruleset> getRulesets() {
        try {
            rulesets = DbManager.getInstance().getRulesets();
        } catch (SQLException e) {
            rulesets = new ArrayList<Ruleset>();
            Messages.err_gets("ruleset");
            e.printStackTrace();        }
        return rulesets;   }
    public String deleteRulesets() {
        // get selected rulesets
        String ids = "";
        String[] arrIds;
        for(Ruleset r: rulesets){
            if(r.isRemoved()){
                ids += r.getId() + ",";          }      }
        if (!ids.isEmpty()) {
            arrIds = ids.split(",");
            try{//delete from db
                DbManager.getInstance().deleteRulesets(arrIds);
                rulesets = DbManager.getInstance().getRulesets(); //update list
                //delete actual files
                for(String s: arrIds)
                    FileManager.deleteDRL(s);
                Messages.suc_delete(arrIds.length+" ruleset(s)");
            } catch (SQLException e){
                Messages.err_delete("Database ruleset(s)");
                e.printStackTrace();
            } catch (IOException e) {
                Messages.err_delete("File(s)");
```

```
            e.printStackTrace();       }  }
    return "";  }
public String getAskDelRul(){ return Messages.getConfirmDel("rulesets");}
/*  * ================= TEMPLATES    */
private ArrayList<Template> templates;
public ArrayList<Template> getTemplates() {
    try {
        templates = DbManager.getInstance().getTemplates();
    } catch (SQLException e) {
        Messages.err_gets("templates");
        e.printStackTrace();
        templates = new ArrayList<Template>();      }
    return templates; }
public String deleteTemplates(){
    // get selected templates
    String names = "";
    String ids = "";
    int ctr = 0;
    for(Template t: templates){
        if(t.isRemoved()){
            names += t.getName() + ",";
            ids += t.getId() + ",";
            ctr++;          }      }
    if (!names.isEmpty()) {
        try{
        //delete from db
        DbManager.getInstance().deleteTemplates(ids.split(","));
        templates = DbManager.getInstance().getTemplates();//update list
        //delete actual files
        for(String name: names.split(","))
            FileManager.deleteTemplate(name);
        Messages.suc_delete(ctr+" template(s)");
        } catch (SQLException e){
            Messages.err_delete(" Database template(s)");
            e.printStackTrace();
        } catch (IOException e) {
            Messages.err_delete(" File(s)");
            e.printStackTrace();           }      }
    return "";  }
public String getAskDelTem(){
    return Messages.getConfirmDel("templates");  }
/* ********************** * RULEGROUPS   ************************** */
private ArrayList<Rulegroup> rulegroups;
public ArrayList<Rulegroup> getRulegroups(){
    try{
        rulegroups = DbManager.getInstance().getRulegroups();
    } catch(SQLException e){
        Messages.err_gets("rulegroups");
        e.printStackTrace();
        rulegroups = new ArrayList<Rulegroup>();    }
    return rulegroups;   }
public String getAskDelRg(){
    return "Are you sure you want to delete checked rulegroups?";   }
public String deleteRulegroups(){
    String delRg = "";
    int ctr = 0;
    for(Rulegroup rg: rulegroups){
        if(rg.isRemoved()){
            delRg += rg.getId() + ",";
            ctr++;          }      }
    if(!delRg.isEmpty()){
        try {
            DbManager.getInstance().deleteRulegroups(delRg.split(","));
            Messages.suc_delete(ctr + " rulegroup(s)");
        } catch (SQLException e) {
            Messages.err_delete("rulegroup");
            e.printStackTrace();           }      }
    return "";  }
private Rulegroup newRg;
public Rulegroup getNewRg() {
    if(newRg==null)
        newRg = new Rulegroup();
    return newRg; }
public void setNewRg(Rulegroup newRg) {
    this.newRg = newRg; }
public String saveNewRg(){
    try {
        newRg.save();
        Messages.suc_save("rulegroup");
        rulegroups.add(newRg);
    } catch (SQLException e) {
        Messages.err_add("rulegroup");
        e.printStackTrace();       }
    newRg = new Rulegroup();
    return "";  } }
```

/WEB-INF/classes/beans/Decision.java

```
package beans;
import java.util.ArrayList;
public class Decision {
    private String decision;
    private ArrayList<String> conditions;
    public Decision(){}
    public Decision(String decision, ArrayList<String> conditions){
        this.decision = decision;
        this.conditions = conditions;   }
    public String getDecision() {       return decision;   }
    public void setDecision(String decision) {      this.decision = decision; }
    public ArrayList<String> getConditions() {      return conditions;}
    public void setConditions(ArrayList<String> conditions) {
        this.conditions = conditions;   }  }
```

/WEB-INF/classes/beans/Function.java

```
package beans;
import general.Globals;import java.sql.SQLException;import
java.util.ArrayList;import database.DbManager;
public class Function {
    //db properties
    private int id;
    private int rulesetId;
    private int returntype;
    private String name;
    private String description;
    private String code;
    //other
    private ArrayList<Parameter> params;
    private boolean isAdded = false;
    private boolean isRemoved = false;
    private int tempObjId;
    /*   * CONSTRUCTORS   */
    public Function(){   }
    public Function(boolean isAdded, String name, int tempObjId){
        this.isAdded = isAdded;
        this.name = name;
        this.tempObjId = tempObjId;  }
    public void init() throws SQLException{
        params = DbManager.getInstance().getParameters(id);   }
    public void save(int rulesetId) throws SQLException{
        this.rulesetId = rulesetId;
        id = DbManager.getInstance().addFunction(this);
        if(params!=null && !params.isEmpty())
            for(Parameter p: params)
                p.save(id);   }
    public void update() throws SQLException{
        DbManager.getInstance().updateFunction(this);
        String delParams = "";
        if(params!=null && !params.isEmpty()){
            for(Parameter p: params){
                if(p.isAdded()&& !p.isRemoved())
                    p.save(id);
```

```
            else if(!p.isAdded()){
                if(p.isRemoved())       delParams += p.getId() + ",";
                else     p.update();         }     }
        if(!delParams.isEmpty())
            DbManager.getInstance().deleteParams(delParams.split(",")); }  }
    public void addParameter(){   //param values from object
        params.add(new Parameter(tempObjId)); }
    public void removeThis(){       isRemoved = true;  }
    public String getAskDel(){
        return "Some rules might be using this. Are you sure you want to remove
this function?";}
    public String getStringDatatype() throws SQLException{
        return Globals.getDataType(returntype);   }
    /*  * GETTERS & SETTERS   */
    public int getId() {       return id;  }
    public void setId(int id) {     this.id = id;  }
    public int getRulesetId() {       return rulesetId;  }
    public void setRulesetId(int rulesetId) {
        this.rulesetId = rulesetId;}
    public int getReturntype() {     return returntype;}
    public void setReturntype(int returntype) {
        this.returntype = returntype;   }
    public String getName() {   return name;   }
    public void setName(String name) {   this.name = name;  }
    public String getDescription() {   return description;   }
    public void setDescription(String description) {
        this.description = description; }
    public String getCode() {     return code; }
    public void setCode(String code) {     this.code = code;}
    public ArrayList<Parameter> getParams() {
        if(params==null)      params = new ArrayList<Parameter>();
        return params;}
    public void setParams(ArrayList<Parameter> params) {
        this.params = params;   }
    public boolean isAdded() {     return isAdded;   }
    public boolean isRemoved() {     return isRemoved;  }  }
```

/WEB-INF/classes/beans/InputForm.java

```
package beans;
import general.Globals;import java.sql.SQLException;import
java.util.ArrayList;import javax.faces.model.SelectItem;
import database.DbManager;
public class InputForm {
    //database columns
    private int rulesetId;
    private int attrId;
    private int objId;
    private String question="";
    private int inputType;
    private String choices="";
    private boolean status = true;
    //helpers
    private String value="";
    private String[] values;
    private Attribute attribute;
    private ArrayList<SelectItem> selectItems;
    /*  * CONSTRUCTORS   */
    public InputForm(){ }
    public InputForm(int rulesetId, Attribute a){
        this.rulesetId = rulesetId;
        this.attribute = a;
        this.attrId = a.getId();
        this.objId = a.getObjId(); }
    /* * METHODS     */
    public void save() throws SQLException{
        DbManager.getInstance().addForm(this);  }
    public void update() throws SQLException{
        DbManager.getInstance().updateForm(this); }
    /*  * INPUT TYPE CHECKERS   */
    public boolean isInputText() throws SQLException{
        if(inputType == 0)          return true;
        if("textbox".equalsIgnoreCase(Globals.getInputType(inputType)))
            return true;
        else return false; }
    public boolean isCheckBox() throws SQLException{
        if("checkbox".equalsIgnoreCase(Globals.getInputType(inputType))){
            initSelectItems();
            return true;     }
        else return false; }
    public boolean isMenu() throws SQLException{
        if("menu".equalsIgnoreCase(Globals.getInputType(inputType))){
            initSelectItems();
            return true;     }
        else return false; }
    public boolean isRadioButton() throws SQLException{
        if("radiobutton".equalsIgnoreCase(Globals.getInputType(inputType))){
            initSelectItems();
            return true;     }
        else return false; }
    private void initSelectItems(){
        selectItems = new ArrayList<SelectItem>();
        for(String s: choices.split(";")){
            s = s.trim();
            selectItems.add(new SelectItem(s,s));      }  }
    /*   * GETTERS & SETTERS   */
    public String getStringAttribute(){   return attribute.getName(); }
    public int getRulesetId() {       return rulesetId;  }
    public void setRulesetId(int ruleId) {      this.rulesetId = ruleId; }
    public int getAttrId() {      return attrId;}
    public void setAttrId(int attrId) {      this.attrId = attrId;   }
    public String getQuestion() {      return question;  }
    public void setQuestion(String question) {     this.question = question;}
    public int getInputType() {       return inputType;}
    public void setInputType(int inputType) {   this.inputType = inputType;}
    public String getChoices() {    return choices;}
    public boolean isStatus() {      return status;   }
    public void setStatus(boolean status) {  this.status = status;}
    public void setChoices(String choices) {    this.choices = choices;   }
    public String getValue() {       return value;   }
    public void setValue(String value) {       this.value = value;  }
    public String[] getValues() {    return values;  }
    public void setValues(String[] values) {      this.values = values;   }
    public Attribute getAttribute() {       return attribute;   }
    public void setAttribute(Attribute attribute) {
        this.attribute = attribute;
        this.attrId = attribute.getId();   }
    public ArrayList<SelectItem> getSelectItems(){ return selectItems;  }
    public int getObjId(){       return objId;}
    public void setObjId(int objId){       this.objId = objId;}  }
```

/WEB-INF/classes/beans/ObjectBean.java

```
package beans;
import general.Globals;import general.Utils;import general.Messages;
import general.Validator;import java.io.IOException;import
java.sql.SQLException;import java.util.ArrayList;import
javax.faces.model.SelectItem;import database.DbManager;
import drools.main.ClassManager;import drools.main.REManager;
public class ObjectBean {
    // database columns
    private int id;
    private String name;
    private String description;
    private String created_by;
    private String date_created;
    private String modified_by;
    private String date_modified;
    // helpers
    private ArrayList<Attribute> attrs;
    private String delAttrId = "";
    private boolean editAttr = false;
```

```
    private boolean removed = false;
    private ArrayList<String> checker;
    /*  * CONSTRUCTOR */
    public ObjectBean() { }
    /** * Initialize attributes of the object    */
    public void init() {
        try {
            attrs = DbManager.getInstance().getAttributes(id);
        } catch (SQLException e) {
            Messages.err_gets("Object");
            e.printStackTrace();        }   }
    /** * Save new object *  * @return navigation case    */
    public String save() {
        if (isValid()){
            try {
                // save to database
                name = name.trim().replaceAll(" ", "_");
                created_by = Utils.getCurrentUser();
                date_created = Utils.getCurrentDate();
                id = DbManager.getInstance().addObject(this);
                for (Attribute a : attrs) {   // save attributes
                    if (!a.isRemoved())          a.save(id);              }
                Globals.initHashObjAttrs();
                Messages.suc_save("Object");
                editAttr = false;
                return "view"+PageManager.OBJECTS;
            } catch (SQLException e) {
                if (e.getMessage().contains("Duplicate entry"))
                    Messages.err_duplicate("Object");
                else{
                    Messages.err_add("object");
                    e.printStackTrace();              }          }       }
        return "";  }
    /** * save changes to object    */
    public String update() {
        if (isValid()){
            try {
                if(editAttr){   //attributes are updated
                    for(Attribute a: attrs){
                        if(a.isAdded() && !a.isRemoved())
                            a.save(id);
                        else if(!a.isAdded()){
                            if(a.isRemoved())//get ids to delete
                                delAttrId += a.getId()+",";
                            else a.update();                        }                }
                    if (!delAttrId.isEmpty()) { //delete removed attributes
                        DbManager.getInstance().deleteAttributes(delAttrId.split(","));
                        REManager.updatePublishedRuleset();
                        //update list of rules in case there are automatically delete
attributes     }
                    Globals.initHashObjAttrs(); //update changes    }
                modified_by = Utils.getCurrentUser();
                date_modified = Utils.getCurrentDate();
                DbManager.getInstance().updateObject(this);
                Messages.suc_update("Object");
                editAttr = false;
                return "view"+PageManager.OBJECTS;

            } catch (SQLException e) {
                Messages.err_update("Object details");
                e.printStackTrace();           }       }
        return "";  }
    /*  * VALIDATIONS */
    public boolean isValid() {
        return (Validator.isNameValid(name, "Class name") && isAttrValid()
                && createFile()); }
    private boolean isAttrValid() {
        checker = new ArrayList<String>();    //for duplicate names
        int attrCtr = 0;
        //count attributes and check names
        if (attrs != null && !attrs.isEmpty()) {
            System.out.println("attrs.");
            for (Attribute a : attrs) {
                if (!a.isRemoved()) {
                    if (checker.contains(a.getVariable())) {
                        Messages.err_duplicate("Attribute");
                        return false;                }
                    if (!Validator.isNameValid(a.getVariable().trim(), "Attribute
variable"))
                        return false;
                    //check if variable starts with capital letter,change to lowercase
                    if(Character.isUpperCase(a.getVariable().charAt(0))){
                        String temp = a.getVariable();
                        char c = temp.charAt(0);
                        //cast char to string
                        a.setVariable(temp.replaceFirst(c+"",
Character.toLowerCase(c)+""));                    }
                    attrCtr++;
                } else
                    attrCtr--;          }
            if (attrCtr < 0) {
                Messages.err_atLeastOne("Objects", "attribute");
                return false;        }      }
        return true;  }
    private boolean createFile() {
        name = name.trim().replaceAll(" ", "_");
        name = Validator.toUpperFirst(name);
        ClassManager cm = new ClassManager();
        try {
            cm.init(name, id, attrs);
            if(cm.writeClass()>0){
                Messages.errorCustom("Error compiling object.");
                return false;        }
        } catch (SQLException e) {
            Messages.err_writeFile("class");
            e.printStackTrace();
            return false;
        } catch (IOException e){
            Messages.err_writeFile("class");
            return false;    }
        return true;  }
    /*  * EDIT DETAILS */
    public String editData(){
        editAttr = true;
        return "add"+PageManager.OBJECTS; }
    public String getTitle(){
        if(editAttr)
            return "Update Object";
        else return "Create New Object";   }
    /** * Add attribute to context    */
    public void addAttribute() {
        attrs.add(new Attribute(id, true));    //true for added   }
    public String getAskDelAtt() {
        return Messages.getConfirmRem("attribute");   }
    public String getAskDelMet() {
        return Messages.getConfirmRem("method");}
    /*  * SELECT ITEMS   */
    public ArrayList<SelectItem> getSelectDataTypes() throws SQLException{
        return Globals.getSelectDataTypes();    }
    /*  * GETTERS & SETTERS   */
    public int getId() {        return id; }
    public void setId(int id) {     this.id = id;  }
    public String getName() {       return name;   }
    public void setName(String name) {       this.name = name;  }
    public String getDescription() {       return description;   }
    public void setDescription(String description) {
        this.description = description; }
    public String getCreated_by() {      return created_by;  }
    public void setCreated_by(String created_by) {
        this.created_by = created_by;}
```

```
    public String getDate_created() {       return date_created;   }
    public void setDate_created(String date_created) {
        this.date_created = date_created; }
    public String getModified_by() {     return modified_by; }
    public void setModified_by(String modified_by) {
        this.modified_by = modified_by; }
    public String getDate_modified() {     return date_modified; }
    public void setDate_modified(String date_modified) {
        this.date_modified = date_modified;  }
    public ArrayList<Attribute> getAttrs() {
        if(attrs==null){
            attrs = new ArrayList<Attribute>();
            attrs.add(new Attribute());      }
        return attrs; }
    public void setAttrs(ArrayList<Attribute> attrs) {    this.attrs = attrs;  }
    public boolean isEditAttr() {    return editAttr; }
    public boolean isRemoved() {    return removed;  }
    public void setRemoved(boolean removed){ this.removed = removed;}}
```

/WEB-INF/classes/beans/Output.java

```
package beans;
import java.util.ArrayList;
public class Output {
    private String object;
    private String attribute;
    private String value;
    private ArrayList<String> conditions;
    /* *CONSTRUCTOR   */
    public Output(){   }
    public Output(String object, String attribute, String value, ArrayList<String>
conditions){
        this.object = object;
        this.attribute = attribute;
        this.value = value;
        this.conditions = conditions;   }
    /*  * GETTERS & SETTERS   */
    public String getObject() {      return object;   }
    public void setObject(String object) {     this.object = object; }
    public String getAttribute() {       return attribute;   }
    public void setAttribute(String attribute) {       this.attribute = attribute;  }
    public String getValue() {       return value;   }
    public void setValue(String value) {       this.value = value;  }
    public ArrayList<String> getConditions() {     return conditions;}
    public void setConditions(ArrayList<String> conditions) {
        this.conditions = conditions;   }  }
```

/WEB-INF/classes/beans/PageManager.java

```
package beans;
import java.sql.SQLException;import javax.faces.context.FacesContext;
import javax.servlet.http.HttpSession;import general.Config;
public class PageManager {
    public static final String HOME = "home";
    public static final String RULES = "rules";
    public static final String RULEGROUPS = "rulegroups";
    public static final String OBJECTS = "objects";
    public static final String SETTINGS = "settings";
    public static final String USERS = "users";
    public static final String TEMPLATES = "templates";
    public static final String RESULT = "result";
    public static final String FORM = "form";
    public static final String ACCOUNT = "account";
    public static final String CHANGEPWD = "changepwd";
    public static final String CHANGEQA = "changeqa";
    public static final String PWDFORGOT = "pwdforgot";
    public static final String LOGIN = "login";
    public PageManager() {     }
    /*  * NAVIGATION   */
    public String viewHome() {     return HOME; }
    public String viewRules() {     return RULES;}
    public String viewRulegroups(){     return RULEGROUPS;   }
    public String viewObjects() {      return OBJECTS;   }
    public String viewSettings(){      return SETTINGS;  }
    public String viewUsers() {     return USERS;    }
    public String viewTemplates() {      return TEMPLATES;  }
    public String viewResult(){     return RESULT;  }
    public String viewInputForm(){       return FORM;  }
    public String changePwd() {       return CHANGEPWD;}
    public String changeQA(){     return CHANGEQA;   }
    public String viewAccount() throws SQLException {
        //get logged user
        SessionBean sb = (SessionBean) FacesContext.getCurrentInstance()
        .getExternalContext().getSessionMap().get("sessionbean");
        //initialize user details
        User user = new User();
        if(sb!=null)
            user.init(sb.getUsername());
        FacesContext.getCurrentInstance()
                .getExternalContext().getSessionMap().put("user", user);
        return ACCOUNT;  }
    public String logout(){
        HttpSession session = (HttpSession)
    FacesContext.getCurrentInstance().getExternalContext().getSession(false);
    if (session != null) {         session.invalidate();       }
        return LOGIN; }
    // ======== Add - with reset
    public String addUser(){
    FacesContext.getCurrentInstance().getExternalContext().getSessionMap()
            .put("user", new User());
        return "add"+USERS; }
    public String addRuleset() {
    FacesContext.getCurrentInstance().getExternalContext().getSessionMap()
                .put("ruleset", new Ruleset());
        return "add"+RULES; }
    public String addObject() {
    FacesContext.getCurrentInstance().getExternalContext().getSessionMap()
                .put("object", new ObjectBean());
        return "add"+OBJECTS; }
    /*  * SITE PROPERTIES     */
    private String frontTemplate;
    private String backTemplate;
    public String getSiteName() {
        return Config.getInstance().getValue(Config.SITE_NAME); }
    public String getFrontTemplate() {
        frontTemplate = Config.getInstance().getValue(Config.SITE_CSS);
        return frontTemplate + "/" + frontTemplate + ".css" ; }
    public String getBackTemplate() {
        if(backTemplate==null)          init();
        return backTemplate; }
    public String getSiteDesc() {
        return Config.getInstance().getValue(Config.SITE_DESC);  }
    public String getSiteMsg() {
        return Config.getInstance().getValue(Config.SITE_MSG);    }
    public String getSiteMsgNoResult(){
        return Config.getInstance().getValue(Config.SITE_NO_RESULT);  }
    public String getContext(){     return Config.getContext(); }
    public void init(){
        String css = Config.getInstance().getValue(Config.SITE_CSS);
        frontTemplate = css + "/" + css + ".css" ;
        css = Config.getInstance().getValue(Config.BACK_CSS);
        backTemplate =  css +  "/"  + css + ".css" ;   }  }
```

/WEB-INF/classes/beans/Parameter.java

```
package beans;
import general.Globals;import java.sql.SQLException;import
java.util.ArrayList;import javax.faces.model.SelectItem;import
database.DbManager;
public class Parameter {
    //db columns
    private int id;
    private int functionId;
```

```
    private String variable;
    private int objId;
    private int attrId;
    private int seqNo;
    //others
    private boolean isAdded = false;
    private boolean isRemoved = false;
    public Parameter(){ }
    public Parameter(int objId){
       isAdded = true;
       this.objId = objId;}
    public void save(int functionId) throws SQLException{
       this.functionId = functionId;
       DbManager.getInstance().addParameter(this); }
    public void update() throws SQLException{
       DbManager.getInstance().updateParameter(this);}
    public ArrayList<SelectItem> getAttributes() throws SQLException{
       if(objId>0)          return Globals.getSelectAttributes(objId);
       else        return new ArrayList<SelectItem>();  }
    public void removeThis(){      isRemoved = true;  }
       public String getStringObj() throws SQLException{
       return Globals.getObjectName(objId);   }
    public String getStringAtt() throws SQLException{
       return Globals.getAttributeName(attrId);    }
    /*  * GETTERS & SETTERS   */
    public int getId() {       return id;  }
    public void setId(int id) {     this.id = id;  }
    public int getFunctionId() {      return functionId;}
    public void setFunctionId(int functionId) {       this.functionId = functionId;}
    public String getVariable() {       return variable;   }
    public void setVariable(String variable) {       this.variable = variable;   }
    public int getObjId() {      return objId;}
    public void setObjId(int objId) {      this.objId = objId;}
    public int getAttrId() {      return attrId;}
    public void setAttrId(int attrId) {      this.attrId = attrId;   }
    public int getSeqNo() {       return seqNo;  }
    public void setSeqNo(int seqNo) {      this.seqNo = seqNo;   }
    public boolean isAdded() {     return isAdded;   }
    public boolean isRemoved() {     return isRemoved;  }}
```

/WEB-INF/classes/beans/Result.java

```
package beans;
import java.util.ArrayList;
public class Result {
    private ArrayList<String> firedRules = new ArrayList<String>();
    private ArrayList<Decision> decisions = new ArrayList<Decision>();
    private ArrayList<Output> output = new ArrayList<Output>();
    public Result() {  }
    public void addDecision(String decision) {
       decisions.add(new Decision(decision, firedRules));
       firedRules = new ArrayList<String>();}
    public void addRule(String rule) {       firedRules.add(rule);   }
    public void addOutput(String obj, String attr, String value) {
       output.add(new Output(obj, attr, value, firedRules));
       firedRules = new ArrayList<String>();}
    /*  * GETTERS & SETTERS   */
    public ArrayList<Decision> getDecisions() {  return decisions; }
    public void setDecisions(ArrayList<Decision> decisions) {
       this.decisions = decisions;  }
    public ArrayList<String> getFiredRules() {     return firedRules;}
    public void setFiredRules(ArrayList<String> firedRules) {
       this.firedRules = firedRules;   }
    public ArrayList<Output> getOutput() {      return output;  }
    public void setOutput(ArrayList<Output> output) {this.output = output;    }
    public void print() {
       for (Decision d : decisions) {
          System.out.println("decision: " + d.getDecision());
          System.out.println("  reasons:");
          for (String s : d.getConditions())
             System.out.println("   -" + s);      }
       for (Output o : output) {
          System.out.println("output: " + o.getObject() + "," +o.getAttribute()
                + "=" + o.getValue());     }  }  }
```

/WEB-INF/classes/beans/Rule.java

```
package beans;
import general.Messages;import java.sql.SQLException;import
java.util.ArrayList;import database.DbManager;
public class Rule {
    private static DbManager db;
    // database columns
    private int id; // for database
    private int ruleId; // agenda-group
    private int rulesetId;
    private int salience;
    private int agenda;
    private boolean no_loop=true;
    private boolean start=false;
    // helpers
    private ArrayList<Condition> arrCond;
    private ArrayList<Consequence> arrCons;
    private int condCtr = 1; // for sequencing of conditions
    private int consCtr = 1; // for sequencing of consequences
    private boolean added = false;
    private boolean removed = false;
    private String delConsId = "";
    private int startObj;
    /*  * CONSTRUCTORS   */
    public Rule() { }
    /** * called when a new rule is added while writing a ruleset  *
     * @param rulesetId    * @param ruleId * @param added    */
    public Rule(int rulesetId, int ruleId, boolean added, int objId, int agenda) {
       this.rulesetId = rulesetId;
       this.ruleId = ruleId;
       this.added = added;
       startObj = objId;
       this.agenda = agenda;    }
    /*  * METHODS   */
    /** * init values of rule for viewing and updating  *
     * @throws SQLException   */
    public void init() throws SQLException {
       db = DbManager.getInstance();
       arrCond = db.getConditions(id);
       arrCons = db.getConsequences(id);  }
    /** * save rule to the database    *
     * @param rulesetId    *           id of the ruleset where this rule belongs
     * @throws SQLException   */
    public void save(int rulesetId) throws SQLException{
       this.rulesetId = rulesetId;
       if(start)        agenda=0;
       //get id for saving conditions and consequence
       id = DbManager.getInstance().addRule(this);
       // iterate through conditions and save conditions not removed
       for (Condition c : arrCond) {
          if (!c.isRemoved())
             c.save(id);      }
       // iterate through consequences and save each
       for (Consequence c : arrCons) {
          c.save(id);      }  }
    /** * update database for changes in the rule * @throws SQLException
     *  * @throws SQLException   */
    public void updateRule() throws SQLException {
       DbManager db = DbManager.getInstance();
       db.updateRule(this);
       updateConditions();
       updateConsequences(); }
    /** * check changes in conditions when a rule is updated  *
     * @param db  * @throws SQLException   */
    private void updateConditions() throws SQLException {
       String delCondId = "";
```

```
        // iterate through conditions and check for added or removed conditions
        for (Condition c : arrCond) {
            if (c.isRemoved() && !c.isAdded())
                delCondId += (c.getId() + ","); // get id's of condition to be
                                    // deleted
            else {
                if (c.isAdded()) {                c.save(id);
                } else                c.update();          }     }
        // delete condition
        if (!delCondId.isEmpty()) {
            db.deleteCondition(delCondId.split(","));    }  }
    /** * check changes in consequences when a rule is updated
     *  * @param db  * @throws SQLException   */
    private void updateConsequences() throws SQLException{
        // iterate through consequences and check of added and deleted
        // consequence
        for (Consequence c : arrCons) {
            if (c.isAdded())            c.save(id);
            else            c.update();      }
        // delete consequence
        if (!delConsId.isEmpty())     db.deleteConsequence(delConsId.split(","));}
    /*  * EDIT DETAILS */
    /** * add condition to the context     */
    public void addCondition() {
        condCtr++;
        arrCond.add(new Condition(1, id, condCtr, startObj)); // 1 for auto
selecting  // "AND"   }
    /** * add consequence to the context  */
    public void addConsequence() {
        consCtr++;
        arrCons.add(new Consequence(id, true, consCtr, startObj)); // true,
conseq is // added
    }
    /** * remove last consequence from the context     */
    public void removeConsequence() {
        int index = arrCons.size() - 1;
        delConsId += (arrCons.get(index).getId() + ",");
        arrCons.remove(index); }
    /*  * BOOLEAN CHECKERS   */
    /** * when the rule is deleted from the context    */
    public void removeThis() {      removed = true;  }
    /** * checks if consequences can still be removed *
     * @return true if there are more than 1 consequence, otherwise, false    */
    public boolean isAllowRemove() {
        if (arrCons == null || arrCons.size() <= 1)          return false;
        else        return true;  }
    /*  * CONFIRMATION MESSAGES    */
    public String getAskDel() {  return Messages.getConfirmRem("rule");  }
    public String getAskCons() {
        return Messages.getConfirmRem("consequence");  }
    public String getAskCond() {
        return Messages.getConfirmRem("condition"); }
    /*  * GETTERS & SETTERS   */
    public int getId() {       return id; }
    public void setId(int id) {     this.id = id;  }
    public int getRuleId() {    return ruleId;   }
    public void setRuleId(int ruleId) {       this.ruleId = ruleId;  }
    public int getRulesetId() {       return rulesetId;  }
    public void setRulesetId(int rulesetId) {      this.rulesetId = rulesetId;}
    public int getSalience() {     return salience;   }
    public void setSalience(int salience) {    this.salience = salience; }
    public int getAgenda() {      return agenda;}
    public void setAgenda(int agenda) {       this.agenda = agenda;    }
    public boolean isNo_loop() {      return no_loop;   }
    public void setNo_loop(boolean no_loop) {this.no_loop = no_loop;  }
    public boolean isStart() {    return start; }
    public void setStart(boolean start) {       this.start = start;  }
    public ArrayList<Condition> getArrCond() {
        if (arrCond == null) {
            arrCond = new ArrayList<Condition>();
            arrCond.add(new Condition(-1, id, condCtr, startObj));
        }      return arrCond;   }
    public void setArrCond(ArrayList<Condition> arrCond) {
        this.arrCond = arrCond;
        // get max condCtr from current list
        condCtr = arrCond.get(arrCond.size() - 1).getSeqNo();}
    public ArrayList<Consequence> getArrCons() {
        if (arrCons == null) {
            arrCons = new ArrayList<Consequence>();
            arrCons.add(new Consequence(id, true, consCtr, startObj));       }
        return arrCons;   }
    public void setArrCons(ArrayList<Consequence> arrCons) {
        this.arrCons = arrCons;
        // get max consCtr from current list
        consCtr = arrCons.get(arrCons.size() - 1).getSeqNo(); }
    public boolean isShowBool() {
        if (arrCond == null || arrCond.size() <= 1)          return false;
        else        return true;  }
    public boolean isAdded() {     return added;  }
    public boolean isRemoved() {     return removed;  }
    public void RemoveThis() {     removed = true;  }  }
```

/WEB-INF/classes/beans/Rulegroup.java

```
package beans;
import java.sql.SQLException;import database.DbManager;import
general.Utils;
public class Rulegroup {
    private int id;
    private String name;
    private String description;
    private String created_by;
    private String date_created;
    private boolean isRemoved = false;
    public Rulegroup(){ }
    public void save() throws SQLException{
        created_by = Utils.getCurrentUser();
        date_created = Utils.getCurrentDate();
        DbManager.getInstance().addRuleGroup(this); }
    public void removeThis(){       isRemoved = true;  }
    public boolean isRemoved(){     return isRemoved;  }
    public void setRemoved(boolean isRemoved){
        this.isRemoved = isRemoved;   }
    public boolean isDefault(){  return "DEFAULT".equals(name);  }
    /*  * GETTERS & SETTERS   */
    public int getId() {       return id; }
    public void setId(int id) {     this.id = id;  }
    public String getName() {       return name;   }
    public void setName(String name) {       this.name = name;  }
    public String getDescription() {       return description;   }
    public void setDescription(String description) {
        this.description = description; }
    public String getCreated_by() {      return created_by;  }
    public void setCreated_by(String created_by) {
        this.created_by = created_by;}
    public String getDate_created() {       return date_created;   }
    public void setDate_created(String date_created) {
        this.date_created = date_created; }  }
```

/WEB-INF/classes/beans/Ruleset.java

```
package beans;
import general.Config;import general.Globals;import general.Utils;import
general.Messages;import general.Validator;import java.io.IOException;import
java.sql.SQLException;import java.util.ArrayList;import
javax.faces.model.SelectItem;import database.DbManager;import
drools.main.DRLCreator;import drools.main.REManager;
public class Ruleset {
    // database columns
    private int id;
    private String name = "";
    private String description = "";
```

```
    private int type = 1; //sequential or nonsequential
    private String objects = ",";
    private String attributes = ",";
    private boolean formset = false;
    private boolean status = false;
    private String created_by;
    private String date_created;
    private String modified_by;
    private String date_modified;
    private int rulegroup;
    private String conclusion;
    // helpers
    private int[] arrObjects;
    private ArrayList<Rule> arrRules;
    private ArrayList<InputForm> arrlForms;
    private ArrayList<Function> arrFunctions;
    private ArrayList<SelectItem> selectObjects;
    private Result result;
    private ArrayList<SelectItem> selectConseqTypes;
    private ArrayList<SelectItem> selectFunctions;
    private ArrayList<SelectItem> selectAgenda;
    private DRLCreator dc;
    private int ruleCtr;
    private String delIds = "";
    private boolean removed = false;
    private boolean set = false; //if type is set, can start writing rules
    private boolean edit = false;
    private String newFuncName;
    private int agenda;
    /*  * CONSTRUCTOR */
    public Ruleset() {}
    /*  * METHODS   */
    /** * initialize rules   * @throws SQLException   */
    public void init() throws SQLException {
        // initialize rules details, conditions and consequence
        ArrayList<Rule> tempRule = DbManager.getInstance().getRules(id);
        arrRules = new ArrayList<Rule>();
        for (Rule r : tempRule) {
            r.init();
            arrRules.add(r);      }
        initSelectAgenda();
        ruleCtr = arrRules.get(arrRules.size()-1).getRuleId();
        //initialize functions
        ArrayList<Function> tempFunc =
DbManager.getInstance().getFunctions(id);
        arrFunctions = new ArrayList<Function>();
        for(Function f: tempFunc){
            f.init();
            arrFunctions.add(f);    }
        String[] temp2 = objects.split(",");
        arrObjects = new int[temp2.length];
        int i = 0;
        for(String s: temp2){
            if(s!=null && !s.isEmpty() && !s.equals("0")){
                arrObjects[i] = Integer.parseInt(s);
                i++;         }      }  }
    /** * save ruleset details to the database
     *  * @return navigation case  * @throws IOException
     *            when writing DRL file   */
    public String save() {
        try {
            if (isValid()) {
                attributes = getAttrUsed(",", arrRules);
                created_by = Utils.getCurrentUser();
                date_created = Utils.getCurrentDate();
                // get ruleset id
                id = DbManager.getInstance().addRuleset(this);
                //write file
                dc.writeDRLFile(id);
                // automatically add inputforms since method is "save"
                addInputForm(attributes.split(","));
                // save rules
                for (Rule r : arrRules) {    r.save(id);       }
                //save functions
                for(Function f: arrFunctions){     f.save(id);       }
                Messages.suc_save("ruleset");
                return  "view"+PageManager.RULES;          }
        } catch (SQLException e) {
            Messages.err_add("ruleset");
            e.printStackTrace();
        } catch (IOException e) {
            Messages.err_writeFile("ruleset");
            e.printStackTrace();
        } catch (IllegalArgumentException e){
            Messages.errorCustom("Error compiling ruleset.\n" + e.getMessage());
    }
        return "";  }
    /** * update database on changes in the ruleset (including rules, conditions
     * and consequences)  *   * @return navigation case
     * @throws IOException  * @throws SQLException   */
    public String update() {
        try {
            if (isValid()) {
                String tempAtt = "";
                if (set) // start checking from the beginning
                    tempAtt = getAttrUsed(",", arrRules);
                else     // continue checking
                    tempAtt = getAttrUsed(attributes, arrRules);
                // there are changes
                if (!tempAtt.equalsIgnoreCase(attributes)) {
                    updateAttrsUsed(tempAtt);            }
                status = false;  //auto unpublish updated ruleset
                modified_by = Utils.getCurrentUser();
                date_modified = Utils.getCurrentDate();
                DbManager.getInstance().updateRuleset(this);
                //update file
                dc.writeDRLFile(id);
                // save rules
                delIds = "";
                for (Rule r : arrRules) {
                    if(r.isAdded() && !r.isRemoved())              r.save(id);
                    else if(!r.isAdded()){
                        if(r.isRemoved())        delIds += r.getId() + ",";
                        else r.updateRule();               }            }
                // delete removed rules if there are any
                if (!delIds.isEmpty())
                    DbManager.getInstance().deleteRules(delIds.split(","));
                //update functions
                delIds = "";
                for(Function f: arrFunctions){
                    if(f.isAdded() && !f.isRemoved())                     f.save(id);
                    else if(!f.isAdded()){
                        if(f.isRemoved())            delIds += f.getId() + ",";
                        else f.update();          }            }
                if(!delIds.isEmpty())
                    DbManager.getInstance().deleteFunction(delIds.split(","));
                Messages.suc_update("Ruleset");
                REManager.unpublishRuleset(id);
                return "view"+PageManager.RULES;       }
        } catch (SQLException e) {
            Messages.err_update("ruleset");
            e.printStackTrace();
        } catch (IOException e) {
            Messages.err_writeFile("ruleset");
            e.printStackTrace();
        } catch (IllegalArgumentException e){
            Messages.errorCustom("Error compiling ruleset.\n" + e.getMessage());
    }
        return "";  }
    /** * Get current list of attributes and objects used
```

```
     *  * @throws SQLException   */
    private String getAttrUsed(String tempAtt, ArrayList<Rule> rules)
            throws SQLException {
        if (rules != null && !rules.isEmpty()) {
            for (Rule r : rules) { // get objects and attributes used
                for (Condition c : r.getArrCond()) { // iterate through
                    // conditions
                    if (!c.isRemoved()) {
                        if (!tempAtt.contains("," + c.getAttrId() + ","))
                            tempAtt += c.getAttrId() + ",";     }  }
                for (Consequence c : r.getArrCons()) { // iterate through
                    // consequences
                    // only type 3 (modify) will use objects and attributes
                    if (c.getType() == 3) {
                        if (!tempAtt.contains("," + c.getAttrId() + ","))
                            tempAtt += c.getAttrId() + ",";     }  }  }
        return tempAtt;    }
    /** * check for inputforms that has to be added or removed   *
     * @param tempAtt  * @throws SQLException   */
    private void updateAttrsUsed(String tempAtt) throws SQLException {
        String temp = "";
        if (tempAtt.length() > 1)
            temp = tempAtt.substring(0, tempAtt.length() - 1); // remove last
        // comma to     // maintain CSV
        else         temp = tempAtt;
        // check for added attributes
        String diff = attributes.replaceFirst(temp, ""); // attributes - tempAtt
        if (!diff.equals(attributes)) {
            deleteInputForm(diff.split(","));
            // formSet remains 1 since there will be no missing inputForm     }
        // check for deleted attributes
        temp = attributes.substring(0, attributes.length() - 1);
        diff = tempAtt.replaceFirst(temp, ""); // tempAtt - attributes
        if (!diff.equals(tempAtt)) {// there are removed objects
            addInputForm(diff.split(","));
            formset = false;       }
        attributes = tempAtt;   }
    /** * Adds new inputform to database *
     * @param attrIds    * @throws SQLException   */
    private void addInputForm(String[] attrIds) throws SQLException {
        if(arrIForms==null)      arrIForms = new ArrayList<InputForm>();
        InputForm i;
        for (String s : attrIds) {
            if (!s.isEmpty()) {
                i = new InputForm(id,
DbManager.getInstance().getAttribute(Integer.parseInt(s)));
                i.save();
                arrIForms.add(i);        }      }  }
    /** * removes inputform from database
     *  * @param attrIds    * @throws SQLException   */
    private void deleteInputForm(String[] attrIds) throws SQLException {
        for (String s : attrIds) {
            if (!s.isEmpty()) {
                DbManager.getInstance().deleteForm(id, Integer.parseInt(s));
            }      }
        arrIForms = DbManager.getInstance().getForms(id, true);    }
    /**
     * Updates contents of inputForms when the user sets it. It is always update
* since inputforms are saved already upon the creation of the rule
     *  * @return navigation case  * @throws SQLException   */
    public String updateForm() throws SQLException {
        for (InputForm i : arrIForms) {        i.update();       }
        if (!formset){
            formset=true;
            DbManager.getInstance().formset(id, formset); // update db
        }
        Messages.suc_update("Input form");
        return ""; }
    /*  * RULE ENGINE     */
    /** *  * @return navigation case  * @throws SQLException
     * @throws Exception  */
    public String runEngine() throws SQLException {
        try {
            result = (new REManager()).runEngine(id, arrIForms);
            System.out.println("result null?" + (result == null));
            return PageManager.RESULT;
        } catch (Exception e) {
            Messages.errorCustom("Error running the engine.\n" +
e.getMessage());    }
        return "";  }
    /** *   * @return true if after running the rule engine, a decision was made.
     *         Otherwise, false.  */
    public boolean isDecided() {
        if (result.getDecisions().isEmpty() && result.getOutput().isEmpty())
            return false;
        else         return true;  }
    /*   * PUBLISH  */
    /** * When status==false, deploys the rule and sets deployed to 1, and vice
     * versa.   */
    public void publish() {
        try {
            if (!status) {
                DbManager.getInstance().publishRuleset(id, false);
                REManager.unpublishRuleset(id);
                status = false;
                Messages.suc_publish("Ruleset "+name, status);
            } else {
                DbManager.getInstance().publishRuleset(id, true);
                REManager.publishRuleset(this);
                status = true;
                Messages.suc_publish("Ruleset "+name, status);         }
        } catch (SQLException e) {
            Messages.err_status("ruleset");
            e.printStackTrace();        }  }
    public String getAskDeploy() {
        return Messages.getConfirmPub(name, status);   }
    public String editData() {
        set = true;
        edit = true;
        return "add"+PageManager.RULES; }
    /*   * VALIDATION    */
    private boolean isValid() throws SQLException {
        if (!isRulesValid()) {
            Messages.err_atLeastOne("Ruleset", "rule");
            return false;    }
        dc = new DRLCreator(this);
        try {
            dc.writeDRL();
        } catch (SQLException e) {
            Messages.errorCustom("Error compiling ruleset");
            e.printStackTrace();
            return false;
        } catch (Exception e){
            Messages.errorCustom("Error compiling ruleset.\n" + e.getMessage());
            e.printStackTrace();
            return false;    }
        return true;  }
    public boolean isRulesValid() throws SQLException {
        int ctr = 0;    // check arrRules
        if (!arrRules.isEmpty()) {
            for (Rule r : arrRules) {if(r.isRemoved()) ctr++;  }  }
        return (ctr < arrRules.size());  }
    /*   * RULESET EDIT    */
    public void typeSet() throws SQLException{
        if(arrObjects == null || arrObjects.length==0)
            Messages.err_atLeastOne("Ruleset", "object");
        else {
            selectConseqTypes = Globals.getSelectConseqTypes();
            if(type==1){ //nonsequential, jumping to next rule not allowed
                for(SelectItem si: selectConseqTypes){
```

```
            if(si.getValue()==(Integer)4){
               selectConseqTypes.remove(si);
               break;              }          }          }
        set = true;
        initSelectObjects();    }  }
   /** * adds a rule in the context when adding or updating a ruleset    */
   public void addRule() {
      ruleCtr++;
      arrRules.add(new Rule(id, ruleCtr, true, arrObjects[0], agenda));
      initSelectAgenda();
      agenda = 0; }
   public void addFunction(){
      if(Validator.isNameValid(newFuncName, "Function name")){
         if(arrFunctions == null)arrFunctions = new ArrayList<Function>();
         arrFunctions.add(new Function(true, newFuncName, arrObjects[0]));
         if(selectFunctions == null)  initSelectFunctions();
         else selectFunctions.add(new SelectItem(newFuncName,
newFuncName));
         newFuncName = "";      }  }
   public boolean isOutputEmpty(){   return result.getOutput().isEmpty();   }
   public boolean isDecisionEmpty(){ return result.getDecisions().isEmpty();}
   /**    * @return String to show if input form is updated or not    */
   public String getFormSetString() {
      if (formset)  return Messages.getMessage("form_set");
      else return Messages.getMessage("form_unset");   }
   public String getStringType(){
      if(type==1)      return "Non-sequential";
      else return "Sequential"; }
   public String getStringNoDecision(){
      if(conclusion==null || conclusion.isEmpty())
         return Config.getInstance().getValue(Config.SITE_NO_RESULT);
      else return conclusion;   }
   public String getStringRulegroup() throws SQLException{
      return Globals.getRulegroupName(rulegroup); }
   public ArrayList<SelectItem> getAllObjects() throws SQLException{
      return Globals.getSelectObjects(); }
   public ArrayList<SelectItem> getSelectInputTypes() throws SQLException{
      return Globals.getSelectInputTypes();   }
   public ArrayList<SelectItem> getSelectDataTypes() throws SQLException{
      return Globals.getSelectDataTypes();   }
   public ArrayList<SelectItem> getSelectRules(){
      if(selectAgenda==null){  initSelectAgenda();    }
      return selectAgenda;  }
   public void initSelectAgenda(){
      selectAgenda = new ArrayList<SelectItem>();
      ArrayList<Integer> checker = new ArrayList<Integer>();
      for(Rule r: arrRules){
         if(!r.isRemoved() && !r.isStart() && !checker.contains(r.getAgenda())){
            selectAgenda.add(new SelectItem(r.getAgenda(), "Group
"+r.getAgenda()));
            checker.add(r.getAgenda());         }     }  }
   public String getStringObjects() throws NumberFormatException,
SQLException{
      String obj = "";
      for(String s: objects.split(","))
         obj += Globals.getObjectName(Integer.parseInt(s)) + ",";
      return obj;   }
   /*  * GETTERS & SETTERS   */
   public int getId() {       return id; }
   public void setId(int id) {     this.id = id;  }
   public String getName() {       return name;   }
   public void setName(String name) {      this.name = name;  }
   public String getDescription() {      return description;   }
   public void setDescription(String description) {
      this.description = description; }
   public boolean isFormset() {      return formset;}
   public void setFormset(boolean formset) { this.formset = formset;    }
   public String getAttributes() {     return attributes; }
   public void setAttributes(String attributes) {this.attributes = attributes; }
   public String getObjects() {    return objects; }
   public void setObjects(String objects) {     this.objects = objects; }
   public boolean isStatus() {     return status;   }
   public void setStatus(boolean status) {     this.status = status; }
   public int getType() {      return type; }
   public void setType(int type) {    this.type = type;  }
   public String getCreated_by() {      return created_by;  }
   public void setCreated_by(String created_by) {
      this.created_by = created_by;}
   public String getDate_created() {      return date_created;   }
   public void setDate_created(String date_created) {
      this.date_created = date_created; }
   public String getModified_by() { return modified_by; }
   public void setModified_by(String modified_by) {
      this.modified_by = modified_by; }
   public String getDate_modified() {     return date_modified; }
   public void setDate_modified(String date_modified) {
      this.date_modified = date_modified;  }
   public int getRulegroup() {      return rulegroup; }
   public void setRulegroup(int rulegroup) {this.rulegroup = rulegroup;}
   public String getConclusion() {      return conclusion;   }
   public void setConclusion(String conclusion) {
      this.conclusion = conclusion; }
   public int[] getArrObjects(){    return arrObjects;   }
   public void setArrObjects(int[] arrObjects){this.arrObjects = arrObjects;  }
   public ArrayList<Rule> getArrRules() {
      if (arrRules == null) {
         arrRules = new ArrayList<Rule>();
         ruleCtr = 1;
         Rule rule = new Rule(id, ruleCtr, true, arrObjects[0], 0);
         if(type==2)  rule.setStart(true);

         arrRules.add(rule);
         initSelectAgenda();    }
      return arrRules;  }
   public void setArrRules(ArrayList<Rule> rules) throws SQLException {
      System.out.println("setArrRules");
      arrRules = new ArrayList<Rule>();
      for (Rule r : rules) {        r.init();        arrRules.add(r);     }
      //assumes the last element of the array has the max rule id
      ruleCtr = arrRules.get(arrRules.size() - 1).getRuleId();
      initSelectAgenda(); }
   public ArrayList<Function> getArrFunctions() {
      if(arrFunctions==null)        arrFunctions = new ArrayList<Function>();
      return arrFunctions;   }
   public void setArrFunctions(ArrayList<Function> arrFunctions) {
      this.arrFunctions = arrFunctions;   }
   public ArrayList<InputForm> getArrIForms() throws SQLException {
      if (arrIForms==null)
         arrIForms = DbManager.getInstance().getForms(id, true);
      return arrIForms; }
   public void setArrIForms(ArrayList<InputForm> arrIForms) {
      this.arrIForms = arrIForms;}
   public ArrayList<SelectItem> getSelectObjects() throws SQLException{
      if(selectObjects==null)          initSelectObjects();
      return selectObjects;  }
   private void initSelectObjects() throws SQLException{
      selectObjects = new ArrayList<SelectItem>();
      objects = ",";
      for(int i: arrObjects){
         if(i>0){
            selectObjects.add(new SelectItem(i, Globals.getObjectName(i)));
            objects += i + ",";       }     }  }
   public ArrayList<SelectItem> getSelectRulegroups() throws SQLException{
      return Globals.getSelectRulegroups();   }
   public Result getResult() {     return result;   }
   public boolean isRemoved() {    return removed;  }
   public void setRemoved(boolean removed) {    this.removed = removed;}
   public ArrayList<SelectItem> getConseqTypes() throws SQLException {
      if(selectConseqTypes==null)         typeSet();
      return selectConseqTypes;}
```

```
    public ArrayList<SelectItem> getSelectOperators() throws SQLException{
        return Globals.getSelectOperator();   }
    public ArrayList<SelectItem> getSelectFunctions(){
        if(selectFunctions==null) selectFunctions = new ArrayList<SelectItem>();
        if(selectFunctions.isEmpty()){ initSelectFunctions();     }
        return selectFunctions;   }
    public void initSelectFunctions(){
        selectFunctions = new ArrayList<SelectItem>();
        if(arrFunctions!=null && !arrFunctions.isEmpty()){
            for(Function f: arrFunctions)
                if(!f.isRemoved())
                  selectFunctions.add(new SelectItem(f.getName(), f.getName()));}}
    public boolean isSet(){       return set;   }
    public boolean isEdit(){      return edit;  }
    public String getTitle(){
        if(edit)         return "Update Ruleset";
        else return "Create New Ruleset"; }
    public String getNewFuncName() {    return newFuncName;    }
    public void setNewFuncName(String newFuncName) {
        this.newFuncName = newFuncName;   }
    public int getAgenda() {      return agenda;}
    public void setAgenda(int agenda) {       this.agenda = agenda;    }}
```

/WEB-INF/classes/beans/SessionBean.java

```
package beans;
public class SessionBean {
    private String username;
    private int type;
    public SessionBean(){}
    public boolean isLoggedIn(){
        return (username!=null && !username.isEmpty()); }
    /*  * GETTERS AND SETTERS  */
    public String getUsername() {    return username;}
    public void setUsername(String username) {
        this.username = username;   }
    public int getType() {       return type; }
    public void setType(int type) {    this.type = type;  }}
```

/WEB-INF/classes/beans/SiteConfiguration.java

```
package beans;
import java.io.IOException;import java.sql.SQLException;import
javax.faces.context.FacesContext;import database.DbConn;import
general.Config;import general.Messages;
public class SiteConfiguration {
    //site properties
    private String name;
    private String url;
    private String description;
    private String welcomeMessage;
    private String template;
    private String noResultMessage;
    //database properties
    private String database;
    private String user;
    private String password;
    private String dbPrefix;
    private Config config = Config.getInstance();
    public SiteConfiguration(){     init();   }
    private void init(){
        name = config.getValue(Config.SITE_NAME);
        url = config.getValue(Config.SITE_URL);
        description = config.getValue(Config.SITE_DESC);
        template = config.getValue(Config.SITE_CSS);
        welcomeMessage = config.getValue(Config.SITE_MSG);
        noResultMessage = config.getValue(Config.SITE_NO_RESULT);
        database = config.getValue(Config.DB_URL);
        user = config.getValue(Config.DB_USER);
        password = config.getValue(Config.DB_PASSWORD);
        dbPrefix = config.getValue(Config.DB_PREFIX);  }
    public String updateSite(){
        try {
            config.setValue(Config.SITE_NAME, name);
            config.setValue(Config.SITE_URL, url);
            config.setValue(Config.SITE_DESC, description);
            config.setValue(Config.SITE_CSS, template);
            config.setValue(Config.SITE_MSG, welcomeMessage);
            config.setValue(Config.SITE_NO_RESULT, noResultMessage);
            config.updateConfig();
            Messages.suc_update("Project settings");
        } catch (IOException e) {
            Messages.err_update("Properties file");
            e.printStackTrace();       }
        //update page manager
        PageManager pm = new PageManager();
        pm.init();
    FacesContext.getCurrentInstance().getExternalContext().getSessionMap().
            put("page", pm);
        return "";  }
    public String updateDb(){
        try{
            config.setValue(Config.DB_URL, database);
            config.setValue(Config.DB_PASSWORD, password);
            config.setValue(Config.DB_PREFIX, dbPrefix);
            config.setValue(Config.DB_USER, user);
            config.updateConfig();
            Messages.suc_update("Database settings");
        } catch (IOException e) {
            Messages.err_update("Properties file");
            e.printStackTrace();       }
        //update database
        try {
            DbConn.init();
        } catch (SQLException e) {
            Messages.errorCustom("Error with new database settings.\n " +
e.getMessage());
            e.printStackTrace();       }
        return "";  }
    public String getAskUpdateDB(){
        return "Changes in these parameters might cause problems in the system.
Are you sure you want to save changes?";    }
    /*   * GETTERS & SETTERS   */
    public String getName() {       return name;   }
    public void setName(String name) {       this.name = name;  }
    public String getUrl() {     return url;}
    public void setUrl(String url) {     this.url = url;}
    public String getDescription() {       return description;   }
    public void setDescription(String description) {
        this.description = description;}
    public String getWelcomeMessage() {       return welcomeMessage;   }
    public void setWelcomeMessage(String welcomeMessage) {
        this.welcomeMessage = welcomeMessage;  }
    public String getNoResultMessage() {       return noResultMessage;   }
    public void setNoResultMessage(String noResultMessage) {
        this.noResultMessage = noResultMessage;  }
    public String getTemplate() {      return template;  }
    public void setTemplate(String template) {     this.template = template;}
    public String getDatabase() {      return database;  }
    public void setDatabase(String database) {   this.database = database;  }
    public String getUser() {      return user;  }
    public void setUser(String user) {       this.user = user;  }
    public String getPassword() {     return password;}
    public void setPassword(String password) {  this.password = password;}
    public String getDbPrefix() {       return dbPrefix;   }
    public void setDbPrefix(String dbPrefix) {  this.dbPrefix = dbPrefix; }}
```

/WEB-INF/classes/beans/Template.java

```
package beans;
import org.richfaces.event.UploadEvent;import
org.richfaces.model.UploadItem;import database.DbManager;import
general.Config;import general.FileManager;import general.Messages;
```

```
import general.Utils;import java.io.FileNotFoundException;import
java.io.FileReader;import java.io.IOException;import java.sql.SQLException;
public class Template {
    //db columns
    private int id;
    private String name = "";
    private String username;//user who uploaded the template
    private String date_uploaded;
    private boolean removed = false;
    private boolean used = false;
    public Template() { }
    public String fileUploadListener(UploadEvent event) throws
FileNotFoundException {
        if (event == null) {           return "";     }
        UploadItem item = event.getUploadItem();
        name = item.getFileName();
        try {
            if(name.endsWith(".css"))
                FileManager.writeTemplateCss(name.replace(".css", ""), new
FileReader(item.getFile()));
            else if(name.endsWith(".zip"))
                FileManager.writeTemplateZip(name.replace(".zip", ""),
item.getData());
            else{
                Messages.errorCustom("Files should only be .zip or .css");
                return "";         }
            name = name.substring(0, name.lastIndexOf("."));
            username = Utils.getCurrentUser();
            date_uploaded = Utils.getCurrentDate();
            DbManager.getInstance().addTemplate(this);
            Messages.suc_save("template");       }
        catch (IOException e) {
            Messages.err_writeFile("template");
        } catch (SQLException e) {
            Messages.err_add("template");
            e.printStackTrace();        }
        return "";  }
    public boolean isDefault(){     return name.equals("default");    }
    public boolean isUsed(){     return used;}
    public void setUsed(boolean used){       this.used = used;}
    public String useThis(){
        Config.getInstance().setValue(Config.SITE_CSS, name);
        used = true;
        Messages.suc_update("Template");
        try {
            Config.getInstance().updateConfig();
        } catch (IOException e) {
            Messages.err_prop();
            e.printStackTrace();        }
        return "";  }
    /*  * GETTERS & SETTERS   */
    public int getId() {        return id;  }
    public void setId(int id) {     this.id = id;   }
    public String getName() {       return name;    }
    public void setName(String name) {
        this.name = name;
        if(name.equals(Config.getInstance().getValue(Config.SITE_CSS)))
            used = true;}
    public String getUsername() {     return username;}
    public void setUsername(String username) {this.username = username;}
    public String getDate_uploaded() {     return date_uploaded;}
    public void setDate_uploaded(String date_uploaded) {
        this.date_uploaded = date_uploaded;    }
    public boolean isRemoved() {     return removed;  }
    public void setRemoved(boolean removed) {this.removed = removed;}  }
```

/WEB-INF/classes/beans/User.java

```
package beans;
import java.sql.SQLException;import java.util.ArrayList;import
javax.faces.context.FacesContext;import
avax.faces.validator.ValidatorException;import javax.faces.model.SelectItem;
import database.DbManager;import general.Globals;import
general.Messages;import general.Validator;
public class User {
    private String username;
    private String password;
    private String name;
    private String details;
    private int type;
    private boolean status;
    private String question;
    private String answer;
    // --for change password
    private String pwdCur;
    private String pwdNew;
    private String pwdVer;
    private String ansVer;
    // for deleting
    private boolean removed = false;
    /*   * CONSTRUCTOR */
    public User() { }
    /*  * METHODS   */
    public void init(String username) {
        System.out.println("User.init");
        try {
            User user = DbManager.getInstance().getUser(username);
            this.username = user.getUsername();
            password = user.getPassword();
            name = user.getName();
            details = user.getDetails();
            type = user.getType();
            question = user.getQuestion();
            answer = user.getAnswer();
        } catch (SQLException e) {
            Messages.err_get("user");
            throw new ValidatorException(null);      }  }
    public String add() {
        try {
            if(Validator.isNameValid(username, "Username")){
                DbManager.getInstance().addUser(this);
                Messages.suc_save("user");
                return PageManager.USERS;
            }
        } catch (SQLException e) {
            if (e.getMessage().contains("Duplicate entry"))
                Messages.err_duplicate("Username");
            else        Messages.errorCustom(e.getMessage());     }
        return "";  }
    public String update(){
        try {
            if(Validator.isNameValid(username, "Username")){
                DbManager.getInstance().updateUser(this);
                Messages.suc_update("user");
                return PageManager.USERS;          }
        } catch (SQLException e) {
            if (e.getMessage().contains("Duplicate entry"))
                Messages.err_duplicate("Username");
            else           Messages.errorCustom(e.getMessage());     }
        return "";  }
    public String login() {
        try {
            User user = DbManager.getInstance().isUserValid(username,
password);
            if(user == null){
                Messages.err_login();       }
            else if(!user.isStatus()){
                Messages.err_userDisabled();          }
            else {
                //init session
                SessionBean sb = new SessionBean();
```

```
            sb.setUsername(username);
            sb.setType(user.getType());
            //put user to session
            FacesContext.getCurrentInstance().getExternalContext().
                    getSessionMap().put("sessionbean", sb);
            return PageManager.HOME;        }
        } catch (SQLException e) {
            Messages.err_get("user");    }
        return ""; }
    public String changePwd() {
        if (!pwdCur.equals(password) || !pwdNew.equals(pwdVer)) {
            Messages.err_pwdChange();
            return PageManager.CHANGEPWD;
        } else {
            try {
                password = pwdNew;
                DbManager.getInstance().changePwd(username, password);
                Messages.suc_update("Password");
                return PageManager.ACCOUNT;
            } catch (SQLException e) {
                Messages.err_update("Password");
                e.printStackTrace();          }     }
        return ""; }
    public String changeQA(){
        try {
                DbManager.getInstance().changeQA(question, answer, username);
                Messages.suc_update("Security question");
                return PageManager.ACCOUNT;
        } catch (SQLException e) {
                Messages.err_update("Security question");
                e.printStackTrace();
        }       return "";        }
    public String pwdForgot() throws SQLException{
        User u = DbManager.getInstance().getUser(username);
        question = u.getQuestion();
        answer = u.getAnswer();
        return PageManager.PWDFORGOT;}
    public String checkForgot() throws SQLException{
        if(!answer.equalsIgnoreCase(ansVer)){
            Messages.errorCustom("Answer invalid.");       }
        else{
            DbManager.getInstance().changePwd(username, username);
            Messages.successCustom("Password set the same as your username.
\n Please update your password immediately after logging in.");}
        return ""; }
    public String getStringType(){
        try {
            return Globals.getUserType(type);
        } catch (SQLException e) {
            Messages.err_gets("user types");
            e.printStackTrace();       }       return ""; }
    public String getStringStatus(){
        if(!status)        return "Enable";
        else return "Disable";  }
    public void activate(){
        try {
            if (!status) {
                DbManager.getInstance().activateUser(username, false);
                status = false;
                Messages.successCustom("User account " + username + "
deactivated.");
            } else {
                DbManager.getInstance().activateUser(username, true);
                status = true;
                Messages.successCustom("User account " + username + "
activated.");        }
        } catch (SQLException e) {
            Messages.errorCustom("Error in (de)activating user.");    }          }
    public String getAskStatus(){      return getStringStatus() + " user?";}
    public boolean isQASet(){
        return (!question.isEmpty() && !answer.isEmpty());   }
    public boolean isUserAdmin(){
        return username.equals("admin")?true:false;}
    public ArrayList<SelectItem> getSelectUserTypes() throws SQLException{
        return Globals.getSelectUserTypes();    }
    /*  * GETTERS & SETTERS   */
    public String getUsername() {     return username;}
    public void setUsername(String username) { this.username = username;}
    public String getPassword() {     return password; }
    public void setPassword(String password) {  this.password = password; }
    public String getName() {       return name;   }
    public void setName(String name) {       this.name = name;  }
    public String getDetails() {       return details;  }
    public void setDetails(String details) {    this.details = details;   }
    public int getType() {       return type; }
    public void setType(int type) {     this.type = type;  }
    public boolean isStatus() {      return status;   }
    public void setStatus(boolean status) {       this.status = status;}
    public String getQuestion() {       return question;   }
    public void setQuestion(String question) { this.question = question;}
    public String getAnswer() {     return answer;}
    public void setAnswer(String answer) {       this.answer = answer;}
    public String getPwdCur() {    return pwdCur;    }
    public void setPwdCur(String pwdCur) {    this.pwdCur = pwdCur;   }
    public String getPwdNew() {       return pwdNew;  }
    public void setPwdNew(String pwdNew) {     this.pwdNew = pwdNew;}
    public String getPwdVer() {    return pwdVer;}
    public void setPwdVer(String pwdVer) {     this.pwdVer = pwdVer;   }
    public String getAnsVer() {      return ansVer;}
    public void setAnsVer(String ansVer) {       this.ansVer = ansVer; }
    public boolean isRemoved() {     return removed;  }
    public void setRemoved(boolean removed) {this.removed = removed;    }}
```

/WEB-INF/classes/Database/DbConn.java

```
package database;
import general.Config;import java.io.IOException;import java.sql.Connection;I
mport java.sql.DriverManager;import java.sql.SQLException;
/** * Creates a database connection using the parameters specified in config
 * @author gessie * */
public class DbConn {
    private static Connection instance;
    /** * Constructor - set to private since this will not be used by other
     * classes. An instance is created only once.  */
    private DbConn() { }
    /** * Returns a database connection instance to avoid creating numerous
connections * @return sql Connection   * @throws SQLException
     * @throws IOException  */
    public static Connection getInstance() throws SQLException {
        if (instance == null)        init();
        return instance;   }
    /** * Initializes database connections using the values obtained from config
     * @throws SQLException   * @throws IOException  */
    public static void init() throws SQLException{
        //get property values
        Config c = Config.getInstance();
        String db = c.getValue(Config.DB_URL);
        String user = c.getValue(Config.DB_USER);
        String password = c.getValue(Config.DB_PASSWORD);
        // get database connection
        instance = DriverManager.getConnection(db, user, password);   }}
```

/WEB-INF/classes/database/DbManager.java

```
package database;
import general.Config;import java.io.IOException;import java.sql.Connection;
import java.sql.PreparedStatement;import java.sql.ResultSet;import
java.sql.SQLException;import java.sql.Statement;import
java.util.ArrayList;import java.util.HashMap;import
javax.faces.model.SelectItem;import beans.Attribute;import beans.Condition;
import beans.Consequence;import beans.Function;import beans.InputForm;
import beans.ObjectBean;import beans.Parameter;import beans.Rule;
```

```
import beans.Rulegroup;import beans.Ruleset;import beans.Template;
import beans.User;
/** * Handles all database processes, from creating connections to running
queries * and retrieving results. *  * @author gessie *  */
public class DbManager {
    private static DbManager instance;
    private static String dbPrefix;
    private static Connection conn;
    private DbManager() {}
    /** * Returns a DbManager instance to avoid several initializations of
     * DbManager   *  * @return DbManager instance
     * @throws SQLException   * @throws IOException  */
    public static DbManager getInstance() throws SQLException{
        if (instance == null)         init();
        return instance;   }
    /** * Initializes DbManager connection and get db prefix from config
     * @throws SQLException   * @throws IOException
     * @throws SQLException   */
    private static void init() throws SQLException {
        // get db prefix
        dbPrefix = Config.getInstance().getValue(Config.DB_PREFIX);
        instance = new DbManager();
        instance.getConnection();  }
    /** * Sets DbManager database connection using DbConn class
     * @throws SQLException   * @throws IOException
     * @throws SQLException   */
    private void getConnection() throws SQLException{
        conn = DbConn.getInstance();   }
    /*  * ***************************************
     *  * DATABASE PROCESSES. The following methods are used when
accessing the   * database from different sources and processes
     *  * *************************************** */
    /*  * ################LOGIN#################   */
    /** * Returns true if there is an entry from the database with the matching
     * username and password  * @throws SQLException   */
    public User isUserValid(String username, String password) throws
SQLException {
        User user = null;
            PreparedStatement pstmt = conn.prepareStatement("SELECT
username, status, type FROM "
                    + dbPrefix + "users WHERE username=? AND password=?");
            try {
                pstmt.setString(1, username);
                pstmt.setString(2, password);
                ResultSet rs = pstmt.executeQuery();
                try {
                    if (rs.next()){ // resultset not empty
                        user = new User();
                        user.setUsername(rs.getString("username"));
                        user.setStatus(rs.getBoolean("status"));
                        user.setType(rs.getInt("type"));
                    }      } finally {      rs.close();           }
            } finally {              pstmt.close();          }
        return user; }
    /*  ######################USER####################    */
    /** * Creates a new user  *  * @param username  * @param password
     * @param name * @param position  * @throws SQLException   */
    public void addUser(User user) throws SQLException {
        PreparedStatement pstmt = conn
                .prepareStatement("INSERT INTO "
                        + dbPrefix
                        + "users(username, password, name, details, type, status)
VALUES(?,?,?,?,?,?)");
        try {
            pstmt.setString(1, user.getUsername());
            pstmt.setString(2, user.getPassword());
            pstmt.setString(3, user.getName());
            pstmt.setString(4, user.getDetails());
            pstmt.setInt(5, user.getType());
            pstmt.setBoolean(6, user.isStatus());
            pstmt.executeUpdate();
        } finally {
            pstmt.close();      }  }
    /** * Get user details    *  * @param username  * @return
     * @throws SQLException   */
    public User getUser(String username) throws SQLException{
        User user = new User();
            PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM
"
                    + dbPrefix + "users WHERE username=?");
            try {
                pstmt.setString(1, username);
                ResultSet rs = pstmt.executeQuery();
                try {
                    if (rs.next()) {
                        user.setUsername(username);
                        user.setName(rs.getString("name"));
                        user.setDetails(rs.getString("details"));
                        user.setPassword(rs.getString("password"));
                        user.setType(rs.getInt("type"));
                        user.setStatus(rs.getBoolean("status"));
                        user.setQuestion(rs.getString("question"));
                        user.setAnswer(rs.getString("answer"));
                    }          } finally {
                    rs.close();       }
            } finally {      pstmt.close();          }
        return user; }
    /** * Gets the list of users (other than the admin); function can be called
     * only by the admin     *  * @return ArrayList<Login>
     * @throws SQLException   */
    public ArrayList<User> getUsers() throws SQLException{
        ArrayList<User> users = new ArrayList<User>();
        PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM "
                + dbPrefix + "users");
        try {
            ResultSet rs = pstmt.executeQuery();
            try {
                // insert returned values to the arraylist
                User user;
                while (rs.next()) {
                    user = new User();
                    user.setUsername(rs.getString("username"));
                    user.setName(rs.getString("name"));
                    user.setDetails(rs.getString("details"));
                    user.setType(rs.getInt("type"));
                    user.setStatus(rs.getBoolean("status"));
                    users.add(user);
                }  } finally {
                rs.close();    }
        } finally {          pstmt.close();      }
        return users;   }
    /** * Saves changes to user data using the given username
     *  * @param username  * @param password   * @param name
     * @param position  * @throws SQLException   */
    public void updateUser(User user) throws SQLException{
        PreparedStatement pstmt = conn.prepareStatement("UPDATE " +
dbPrefix  + "users SET password=?, name=?, details=?, type=? WHERE
username=?");
        try {
            pstmt.setString(1, user.getPassword());
            pstmt.setString(2, user.getName());
            pstmt.setString(3, user.getDetails());
            pstmt.setInt(4, user.getType());
            pstmt.setString(5, user.getUsername());
            pstmt.executeUpdate();
        } finally {          pstmt.close();      }  }
    public void changePwd(String username, String password) throws
SQLException{
        PreparedStatement pstmt = conn.prepareStatement("UPDATE " +
dbPrefix + "users SET password=? WHERE username=?");
```

```
        try {
            pstmt.setString(1, password);
            pstmt.setString(2, username);
            pstmt.executeUpdate();
        } finally {         pstmt.close();     }  }
    public void activateUser(String username, boolean activated) throws
SQLException{
        PreparedStatement pstmt = conn.prepareStatement("UPDATE " +
dbPrefix  + "users SET status=? WHERE username=?");
        try {
            pstmt.setBoolean(1, activated);
            pstmt.setString(2, username);
            pstmt.executeUpdate();
        } finally {         pstmt.close();     }         }
    public void changeQA(String q, String a, String u) throws SQLException{
        PreparedStatement pstmt = conn.prepareStatement("UPDATE " +
dbPrefix  + "users SET question=?, answer=? WHERE username=?");
        try {
            pstmt.setString(1, q);
            pstmt.setString(2, a);
            pstmt.setString(3, u);
            pstmt.executeUpdate();
        } finally {         pstmt.close();     }                }
    /** * Deletes users from the database, other than the admin
     *  * @param username  * @throws SQLException   */
    public void deleteUsers(String[] usernames) throws SQLException {
        String query = "DELETE FROM " + dbPrefix + "users WHERE ";
        for (int i = usernames.length - 1; i > 0; i--) {
            query += "username='" + usernames[i] + "' || ";     }
        query += "username='" + usernames[0] + "'";
        Statement stmt = conn.createStatement();
        try {         stmt.executeUpdate(query);
        } finally {      stmt.close();    }  }
    /* ################OBJECTS################## */
    /** * Creates new object  *  * @param username  * @param password
     * @param name * @param position  * @throws SQLException
     * @throws SQLException   */
    public int addObject(ObjectBean ob) throws SQLException{
        PreparedStatement pstmt = conn.prepareStatement("INSERT INTO "
              + dbPrefix + "object(name, description, created_by, date_created)
VALUES(?,?,?,?)");
        try {
            pstmt.setString(1, ob.getName());
            pstmt.setString(2, ob.getDescription());
            pstmt.setString(3, ob.getCreated_by());
            pstmt.setString(4, ob.getDate_created());
            pstmt.executeUpdate();
        } finally {         pstmt.close();     }
        return getRecentId("object");  }
    public ObjectBean getObject(int id) throws SQLException {
        ObjectBean ob = new ObjectBean();
        PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM "
              + dbPrefix + "object WHERE id=?");
        try {
            pstmt.setInt(1, id);
            ResultSet rs = pstmt.executeQuery();
            try {
                // get returned id
                if (rs.next()) {
                    ob.setId(rs.getInt("id"));
                    ob.setName(rs.getString("name"));
                    ob.setDescription(rs.getString("description"));
                    ob.setCreated_by(rs.getString("created_by"));
                    ob.setDate_created(rs.getString("date_created"));
                }
            } finally {             rs.close();       }
        } finally {         pstmt.close();     }
        return ob;}
    /** * Gets the list of objects  *  * @return ArrayList<ObjectBean>
     * @throws SQLException   */
    public ArrayList<ObjectBean> getObjects() throws SQLException {
        ArrayList<ObjectBean> objs = new ArrayList<ObjectBean>();
        PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM "
              + dbPrefix + "object");
        try {
            ResultSet rs = pstmt.executeQuery();
            try {
                // insert returned values to the arraylist
                ObjectBean obj;
                while (rs.next()) {
                    obj = new ObjectBean();
                    obj.setId(rs.getInt("id"));
                    obj.setName(rs.getString("name"));
                    obj.setDescription(rs.getString("description"));
                    obj.setCreated_by(rs.getString("created_by"));
                    obj.setDate_created(rs.getString("date_created"));
                    obj.setModified_by(rs.getString("modified_by"));
                    obj.setDate_modified(rs.getString("date_modified"));
                    objs.add(obj);
                }
            } finally {             rs.close();       }
        } finally {         pstmt.close();     }
        return objs;  }
    public HashMap<Integer, String> getObjectHash() throws SQLException {
        HashMap<Integer, String> hash = new HashMap<Integer, String>();
        PreparedStatement pstmt = conn.prepareStatement("SELECT id, name
FROM "   + dbPrefix + "object");
        try {
            ResultSet rs = pstmt.executeQuery();
            try {     // get returned id
                while (rs.next()) { hash.put(rs.getInt("id"), rs.getString("name"));   }
            } finally {             rs.close();       }
        } finally {         pstmt.close();      }       return hash; }
    public void updateObject(ObjectBean ob) throws SQLException {
        PreparedStatement pstmt = conn
                .prepareStatement("UPDATE "   + dbPrefix    + "object SET
description=?, modified_by=?, date_modified=?, name=? WHERE id=?");
        try {
            pstmt.setString(1, ob.getDescription());
            pstmt.setString(2, ob.getModified_by());
            pstmt.setString(3, ob.getDate_modified());
            pstmt.setString(4, ob.getName());
            pstmt.setInt(5, ob.getId());
            pstmt.executeUpdate();
        } finally {         pstmt.close();     }  }
    /** * Deletes an object *  * @param username  * @throws SQLException*/
    public void deleteObjects(String[] ids) throws SQLException {
        deleteWithCol(ids, "object", "id");
        deleteWithCol(ids, "attribute", "objId");

        //delete rulesets using objects
        String query = "DELETE FROM " + dbPrefix + "ruleset WHERE ";
        for (int i = ids.length - 1; i > 0; i--) {
            if(!ids[i].isEmpty())   query += "objects LIKE '%," + ids[i] + ",%' || ";
        }
        if(!ids[0].isEmpty())  query += "objects LIKE '%," + ids[0] + ",%'";
        Statement stmt = conn.createStatement();
        try {     stmt.executeUpdate(query);
        } finally {         stmt.close();    }          }
    /* ############ATTRIBUTES################  */
    /** * Adds new attributes to objects     *  * @param username
     * @param password   * @param name * @param position
     * @throws SQLException   */
    public void addAttribute(Attribute a) throws SQLException {
        PreparedStatement pstmt = conn
                .prepareStatement("INSERT INTO "  + dbPrefix
+ "attribute(objId, datatype, variable, name, description) VALUES(?,?,?,?,?)");
        try {
            pstmt.setInt(1,a.getObjId());
            pstmt.setInt(2, a.getDatatype());
```

```
            pstmt.setString(3, a.getVariable());
            pstmt.setString(4, a.getName());
            pstmt.setString(5, a.getDescription());
            pstmt.executeUpdate();
        } finally {         pstmt.close();     }  }
    public Attribute getAttribute(int id) throws SQLException {
        Attribute attr = new Attribute();
        PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM "
                + dbPrefix + "attribute WHERE id=?");
        try {
            pstmt.setInt(1, id);
            ResultSet rs = pstmt.executeQuery();
            try {
                // insert returned values to the arraylist
                if (rs.next()) {
                    attr = new Attribute();
                    attr.setId(rs.getInt("id"));
                    attr.setObjId(rs.getInt("objId"));
                    attr.setDatatype(rs.getInt("datatype"));
                    attr.setVariable(rs.getString("variable"));
                    attr.setName(rs.getString("name"));
                    attr.setDescription(rs.getString("description"));
                }         } finally {             rs.close();       }
        } finally {         pstmt.close();     }
        return attr;   }
    /** * Gets the list of attributes of a particular object
     *  * @return ArrayList<Attribue>    * @throws SQLException   */
    public ArrayList<Attribute> getAttributes(int obj) throws SQLException {
        ArrayList<Attribute> attrs = new ArrayList<Attribute>();
        PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM "
                + dbPrefix + "attribute WHERE objId=?");
        try {
            pstmt.setInt(1, obj);
            ResultSet rs = pstmt.executeQuery();
            try {
                // insert returned values to the arraylist
                Attribute attr;
                while (rs.next()) {
                    attr = new Attribute();
                    attr.setId(rs.getInt("id"));
                    attr.setObjId(obj);
                    attr.setDatatype(rs.getInt("datatype"));
                    attr.setVariable(rs.getString("variable"));
                    attr.setName(rs.getString("name"));
                    attr.setDescription(rs.getString("description"));
                    attrs.add(attr);            }
            } finally {             rs.close();       }
        } finally {         pstmt.close();     }
        return attrs; }
    public void updateAttribute(Attribute a) throws SQLException {
        PreparedStatement pstmt = conn
                .prepareStatement("UPDATE "  + dbPrefix+ "attribute SET
datatype=?, variable=?, name=?, description=? WHERE id=?");
        try {
            pstmt.setInt(1, a.getDatatype());
            pstmt.setString(2, a.getVariable());
            pstmt.setString(3, a.getName());
            pstmt.setString(4, a.getDescription());
            pstmt.setInt(5, a.getId());
            pstmt.executeUpdate();
        } finally {         pstmt.close();     }  }
    public void deleteAttributes(String[] ids) throws SQLException {
        deleteWithCol(ids, "attribute", "id");
        //delete rulesets using objects
        String query = "DELETE FROM " + dbPrefix + "ruleset WHERE ";
        for (int i = ids.length - 1; i > 0; i--) {
            if(!ids[i].isEmpty())   query += "attributes LIKE '%," + ids[i] + ",%' || ";}
        if(!ids[0].isEmpty())
            query += "attributes LIKE '%," + ids[0] + ",%'";
        Statement stmt = conn.createStatement();
        try {         stmt.executeUpdate(query);
        } finally {         stmt.close();    }  }
    /*################FUNCTION###################  */
    public int addFunction(Function f) throws SQLException {
        PreparedStatement pstmt = conn
                .prepareStatement("INSERT INTO "+ dbPrefix+ "function(rulesetId,
returntype, name, description, code) VALUES(?,?,?,?,?)");
        try {
            pstmt.setInt(1,f.getRulesetId());
            pstmt.setInt(2, f.getReturntype());
            pstmt.setString(3, f.getName());
            pstmt.setString(4, f.getDescription());
            pstmt.setString(5, f.getCode());
            pstmt.executeUpdate();
        } finally {         pstmt.close();     }
        return getRecentId("function");   }
    public ArrayList<Function> getFunctions(int rulesetId) throws SQLException
{
        ArrayList<Function> functions = new ArrayList<Function>();
        PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM "
                + dbPrefix + "function WHERE rulesetId=?");
        try {
            pstmt.setInt(1, rulesetId);
            ResultSet rs = pstmt.executeQuery();
            try {
                // insert returned values to the arraylist
                Function f;
                while (rs.next()) {
                    f = new Function();
                    f.setId(rs.getInt("id"));
                    f.setRulesetId(rulesetId);
                    f.setReturntype(rs.getInt("returntype"));
                    f.setName(rs.getString("name"));
                    f.setDescription(rs.getString("description"));
                    f.setCode(rs.getString("code"));
                    functions.add(f);             }
            } finally {             rs.close();       }
        } finally {         pstmt.close();     }
        return functions;  }
    public void updateFunction(Function f) throws SQLException {
        PreparedStatement pstmt = conn
                .prepareStatement("UPDATE "+ dbPrefix+ "function SET name=?,
returntype=?, description=?, code=? WHERE id=?");
        try {
            pstmt.setString(1, f.getName());
            pstmt.setInt(2, f.getReturntype());
            pstmt.setString(3, f.getDescription());
            pstmt.setString(4, f.getCode());
            pstmt.setInt(5, f.getId());
            pstmt.executeUpdate();
        } finally {         pstmt.close();     }  }
    public void deleteFunction(String[] ids) throws SQLException {
        deleteWithCol(ids, "function", "id");//delete functions
        deleteWithCol(ids, "parameter", "functionId");   }
    /*#############PARAMETER################   */
    public void addParameter(Parameter p) throws SQLException {
        PreparedStatement pstmt = conn
                .prepareStatement("INSERT INTO "+ dbPrefix+
"parameter(functionId, variable, objId, attrId) VALUES(?,?,?,?)");
        try {
            pstmt.setInt(1,p.getFunctionId());
            pstmt.setString(2, p.getVariable());
            pstmt.setInt(3, p.getObjId());
            pstmt.setInt(4, p.getAttrId());
            pstmt.executeUpdate();
        } finally {         pstmt.close();     }  }
    public ArrayList<Parameter> getParameters(int functionId) throws
SQLException {
        ArrayList<Parameter> params = new ArrayList<Parameter>();
        PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM "
```

```
            + dbPrefix + "parameter WHERE functionId=?");
        try {
            pstmt.setInt(1, functionId);
            ResultSet rs = pstmt.executeQuery();
            try {
                // insert returned values to the arraylist
                Parameter p;
                while (rs.next()) {
                    p = new Parameter();
                    p.setId(rs.getInt("id"));
                    p.setFunctionId(functionId);
                    p.setVariable(rs.getString("variable"));
                    p.setObjId(rs.getInt("objId"));
                    p.setAttrId(rs.getInt("attrId"));
                    params.add(p);
                }
            } finally {             rs.close();       }
        } finally {         pstmt.close();     }
        return params;}
    public void updateParameter(Parameter p) throws SQLException {
        PreparedStatement pstmt = conn
                .prepareStatement("UPDATE "+ dbPrefix+ "parameter SET
variable=?, objId=?, attrId=? WHERE id=?");
        try {
            pstmt.setString(1, p.getVariable());
            pstmt.setInt(2, p.getObjId());
            pstmt.setInt(3, p.getAttrId());
            pstmt.setInt(4, p.getId());
            pstmt.executeUpdate();
        } finally {         pstmt.close();     }  }
    public void deleteParams(String[] ids) throws SQLException {
        deleteWithCol(ids, "parameter", "id");}
    /* ############RULESET################ */
    public int addRuleset(Ruleset r) throws SQLException {
        PreparedStatement pstmt = conn
                .prepareStatement("INSERT INTO "+ dbPrefix+ "ruleset(description,
attributes, objects, name, type, created_by, date_created, rulegroup,
conclusion)" + " VALUES(?,?,?,?,?,?,?,?,?)");
        try {
            pstmt.setString(1, r.getDescription());
            pstmt.setString(2, r.getAttributes());
            pstmt.setString(3, r.getObjects());
            pstmt.setString(4, r.getName());
            pstmt.setInt(5, r.getType());
            pstmt.setString(6, r.getCreated_by());
            pstmt.setString(7, r.getDate_created());
            pstmt.setInt(8, r.getRulegroup());
            pstmt.setString(9, r.getConclusion());
            pstmt.executeUpdate();
        } finally {         pstmt.close();     }
        return getRecentId("ruleset");
    }
    public Ruleset getRuleset(int id) throws SQLException {
        Ruleset r = new Ruleset();
        PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM "
                + dbPrefix + "ruleset WHERE id=?");
        try {
            pstmt.setInt(1, id);
            ResultSet rs = pstmt.executeQuery();
            try {
                // get returned id
                if (rs.next()) {
                    r.setId(id);
                    r.setName(rs.getString("name"));
                    r.setDescription(rs.getString("description"));
                    r.setFormset(rs.getBoolean("formset"));
                    r.setStatus(rs.getBoolean("status"));
                    r.setAttributes(rs.getString("attributes"));
                    r.setObjects(rs.getString("objects"));
                    r.setType(rs.getInt("type"));
                    r.setCreated_by(rs.getString("created_by"));
                    r.setDate_created(rs.getString("date_created"));
                    r.setModified_by(rs.getString("modified_by"));
                    r.setDate_modified(rs.getString("date_modified"));
                    r.setRulegroup(rs.getInt("rulegroup"));
                    r.setConclusion(rs.getString("conclusion"));
                }
            } finally {             rs.close();       }
        } finally {         pstmt.close();     }
        return r;   }
    public ArrayList<Ruleset> getRulesets() throws SQLException {
        ArrayList<Ruleset> rules = new ArrayList<Ruleset>();
        Ruleset r;
        PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM "
                + dbPrefix + "ruleset");
        try {
            ResultSet rs = pstmt.executeQuery();
            try {
                // get returned id
                while (rs.next()) {
                    r = new Ruleset();
                    r.setId(rs.getInt("id"));
                    r.setName(rs.getString("name"));
                    r.setDescription(rs.getString("description"));
                    r.setFormset(rs.getBoolean("formset"));
                    r.setStatus(rs.getBoolean("status"));
                    r.setAttributes(rs.getString("attributes"));
                    r.setObjects(rs.getString("objects"));
                    r.setType(rs.getInt("type"));
                    r.setCreated_by(rs.getString("created_by"));
                    r.setDate_created(rs.getString("date_created"));
                    r.setModified_by(rs.getString("modified_by"));
                    r.setDate_modified(rs.getString("date_modified"));
                    r.setRulegroup(rs.getInt("rulegroup"));
                    r.setConclusion(rs.getString("conclusion"));
                    rules.add(r);
                }
            } finally {             rs.close();       }
        } finally {         pstmt.close();     }
        return rules;}
    public ArrayList<Ruleset> getPublishedRulesets() throws SQLException {
        ArrayList<Ruleset> rules = new ArrayList<Ruleset>();
        Ruleset r;
        PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM "
                + dbPrefix + "ruleset WHERE status=?");
        try {
            pstmt.setBoolean(1, true);
            ResultSet rs = pstmt.executeQuery();
            try {
                // get returned id
                while (rs.next()) {
                    r = new Ruleset();
                    r.setId(rs.getInt("id"));
                    r.setName(rs.getString("name"));
                    r.setDescription(rs.getString("description"));
                    r.setFormset(rs.getBoolean("formset"));
                    r.setStatus(rs.getBoolean("status"));
                    r.setRulegroup(rs.getInt("rulegroup"));
                    rules.add(r);           }
            } finally {             rs.close();       }
        } finally {         pstmt.close();     }
        return rules;}
    public String getObjectsUsed(int ruleId) throws SQLException {
        String ids = "";
        PreparedStatement pstmt = conn.prepareStatement("SELECT objects
FROM "   + dbPrefix + "ruleset WHERE id=?");
        try {
            pstmt.setInt(1, ruleId);
            ResultSet rs = pstmt.executeQuery();
            try { // get returned id
```

```
            if (rs.next()) {   ids = rs.getString("objects");   }
          } finally {            rs.close();       }
       } finally {         pstmt.close();     }
       return ids;   }
   public void updateRuleset(Ruleset r)
         throws SQLException {
       PreparedStatement pstmt = conn
            .prepareStatement("UPDATE "+ dbPrefix+ "ruleset SET
description=?, objects=?, attributes=?, formSet=?, modified_by=?,
date_modified=?, rulegroup=?, status=?, conclusion=? WHERE id=?");
       try {
          pstmt.setString(1, r.getDescription());
          pstmt.setString(2, r.getObjects());
          pstmt.setString(3, r.getAttributes());
          pstmt.setBoolean(4, r.isFormset());
          pstmt.setString(5, r.getModified_by());
          pstmt.setString(6, r.getDate_modified());
          pstmt.setInt(7, r.getRulegroup());
          pstmt.setBoolean(8, r.isStatus());
          pstmt.setString(9, r.getConclusion());
          pstmt.setInt(10, r.getId());
          pstmt.executeUpdate();
       } finally {         pstmt.close();     }  }
   public void formset(int ruleId, boolean value) throws SQLException {
       PreparedStatement pstmt = conn.prepareStatement("UPDATE " +
dbPrefix  + "ruleset SET formSet=? WHERE id=?");
       try {
          pstmt.setBoolean(1, value);
          pstmt.setInt(2, ruleId);
          pstmt.executeUpdate();
       } finally {         pstmt.close();     }  }
   public void deleteRulesets(String[] ids) throws SQLException {
       //delete rules and rule related data
       String tempIds = getIdsWithCol(ids, "rule", "rulesetId");
       if(!tempIds.isEmpty()){
          String[] rules = tempIds.split(",");
          deleteWithCol(rules, "condition", "ruleId");
          deleteWithCol(rules, "consequence", "ruleId");     }
       //delete function params
       tempIds = getIdsWithCol(ids, "function", "rulesetId");
       if(!tempIds.isEmpty())
          deleteWithCol(tempIds.split(","), "parameter", "functionId");
       deleteWithCol(ids, "ruleset", "id");
       deleteWithCol(ids, "rule", "rulesetId");
       deleteWithCol(ids, "inputform", "rulesetId");
       deleteWithCol(ids, "function", "rulesetId"); }

   public void deleteRules(String ids[]) throws SQLException{
       deleteWithCol(ids, "rule", "id");   }
   public void publishRuleset(int rulesetId, boolean status) throws
SQLException{
       PreparedStatement pstmt = conn.prepareStatement("UPDATE " +
dbPrefix+ "ruleset SET status=? WHERE id=?");
       try {
          pstmt.setBoolean(1, status);
          pstmt.setInt(2, rulesetId);
          pstmt.executeUpdate();
       } finally {         pstmt.close();     }  }
   /* ###############RULES##################### */
   public int addRule(Rule rule) throws SQLException {
       PreparedStatement pstmt = conn.prepareStatement("INSERT INTO "
   + dbPrefix + "rule(ruleId, rulesetId, salience, no_loop, start, agenda)
VALUES(?,?,?,?,?,?)");
       try {
          pstmt.setInt(1, rule.getRuleId());
          pstmt.setInt(2, rule.getRulesetId());
          pstmt.setInt(3, rule.getSalience());
          pstmt.setBoolean(4, rule.isNo_loop());
          pstmt.setBoolean(5, rule.isStart());
          pstmt.setInt(6, rule.getAgenda());
          pstmt.executeUpdate();
       } finally {         pstmt.close();     }
       return getRecentId("rule");
   }
   public ArrayList<Rule> getRules(int rulesetId) throws SQLException {
       ArrayList<Rule> rules = new ArrayList<Rule>();
       PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM "
            + dbPrefix + "rule WHERE rulesetId=? ORDER BY ruleId");
       try {
          pstmt.setInt(1, rulesetId);
          ResultSet rs = pstmt.executeQuery();
          try {
             Rule r;
             while (rs.next()) {
                r = new Rule();
                r.setId(rs.getInt("id"));
                r.setRuleId(rs.getInt("ruleId"));
                r.setRulesetId(rs.getInt("rulesetId"));
                r.setSalience(rs.getInt("salience"));
                r.setNo_loop(rs.getBoolean("no_loop"));
                r.setStart(rs.getBoolean("start"));
                r.setAgenda(rs.getInt("agenda"));
                rules.add(r);
             }
          } finally {            rs.close();       }
       } finally {         pstmt.close();     }
       return rules; }
   // only update attributes, NOT rule ids and ruleset ids
   public void updateRule(Rule r) throws SQLException {
       PreparedStatement pstmt = conn.prepareStatement("UPDATE " +
dbPrefix+ "rule SET salience=?, start=?, no_loop=?, agenda=? WHERE id=?");
       try {
          pstmt.setInt(1, r.getSalience());
          pstmt.setBoolean(2, r.isStart());
          pstmt.setBoolean(3, r.isNo_loop());
          pstmt.setInt(4, r.getAgenda());
          pstmt.setInt(5, r.getId());
          pstmt.executeUpdate();
       } finally {         pstmt.close();     }  }
   /*##############CONDITION#################   */
   public void addCondition(Condition c) throws SQLException {
       PreparedStatement pstmt = conn
            .prepareStatement("INSERT INTO "+ dbPrefix+ "condition(ruleId,
objId, attrId, optrId, value, bool, seqNo, type, function)
VALUES(?,?,?,?,?,?,?,?,?)");
       try {
          pstmt.setInt(1, c.getRuleId());
          pstmt.setInt(2, c.getObjId());
          pstmt.setInt(3, c.getAttrId());
          pstmt.setInt(4, c.getOptrId());
          pstmt.setString(5, c.getValue());
          pstmt.setInt(6, c.getBool());
          pstmt.setInt(7, c.getSeqNo());
          pstmt.setInt(8, c.getType());
          pstmt.setString(9, c.getFunction());
          pstmt.executeUpdate();
       } finally {         pstmt.close();     }  }
   public ArrayList<Condition> getConditions(int ruleId) throws SQLException {
       ArrayList<Condition> conditions = new ArrayList<Condition>();
       PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM "
            + dbPrefix + "condition WHERE ruleId=? ORDER BY seqNo");
       try {
          pstmt.setInt(1, ruleId);
          ResultSet rs = pstmt.executeQuery();
          try {
             // get returned id
             Condition con;
             while (rs.next()) {
                con = new Condition();
                con.setId(rs.getInt("id"));
```

```
                con.setRuleId(ruleId);
                con.setAttrId(rs.getInt("attrId"));
                con.setObjId(rs.getInt("objId"));
                con.setOptrId(rs.getInt("optrId"));
                con.setValue(rs.getString("value"));
                con.setBool(rs.getInt("bool"));
                con.setType(rs.getInt("type"));
                con.setSeqNo(rs.getInt("seqNo"));
                con.setAdded(false);
                con.setFunction(rs.getString("function"));
                conditions.add(con);
            }
        } finally {             rs.close();       }
    } finally {         pstmt.close();     }
    return conditions;}
  public void updateCondition(Condition c) throws SQLException {
    PreparedStatement pstmt = conn
          .prepareStatement("UPDATE "+ dbPrefix+ "condition SET objId=?,
attrId=?, optrId=?, value=?, bool=?, type=?, function=? WHERE id=?");
    try {
      pstmt.setInt(1, c.getObjId());
      pstmt.setInt(2, c.getAttrId());
      pstmt.setInt(3, c.getOptrId());
      pstmt.setString(4, c.getValue());
      pstmt.setInt(5, c.getBool());
      pstmt.setInt(6, c.getType());
      pstmt.setString(7, c.getFunction());
      pstmt.setInt(8, c.getId());
      pstmt.executeUpdate();
    } finally {         pstmt.close();     }  }
  public void deleteCondition(String[] ids) throws SQLException {
    deleteWithCol(ids, "condition", "id");  }

  /*###############CONSEQUENCE################# */
  public void addConsequence(Consequence c) throws SQLException {
    PreparedStatement pstmt = conn
        .prepareStatement("INSERT INTO "+ dbPrefix+ "consequence(ruleId,
type, value, objId, attrId, function, seqNo) VALUES(?,?,?,?,?,?,?)");
    try {
      pstmt.setInt(1, c.getRuleId());
      pstmt.setInt(2, c.getType());
      pstmt.setString(3, c.getValue());
      pstmt.setInt(4, c.getObjId());
      pstmt.setInt(5, c.getAttrId());
      pstmt.setString(6, c.getFunction());
      pstmt.setInt(7, c.getSeqNo());
      pstmt.executeUpdate();
    } finally {         pstmt.close();     }  }
  public ArrayList<Consequence> getConsequences(int ruleId)
      throws SQLException {
    ArrayList<Consequence> consequences = new
ArrayList<Consequence>();
    PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM "
        + dbPrefix + "consequence WHERE ruleId=? ORDER BY seqNo");
    try {
      pstmt.setInt(1, ruleId);
      ResultSet rs = pstmt.executeQuery();
      try {  // get returned id
        Consequence con;
        while (rs.next()) {
          con = new Consequence(ruleId, false, rs.getInt("seqNo"),
rs.getInt("objId"));
          con.setId(rs.getInt("id"));
          con.setType(rs.getInt("type"));
          con.setAttrId(rs.getInt("attrId"));
          con.setValue(rs.getString("value"));
          con.setFunction(rs.getString("function"));
          consequences.add(con);          }
      } finally {             rs.close();       }
    } finally {         pstmt.close();     }
    return consequences; }
  public void updateConsequence(Consequence c) throws SQLException {
    PreparedStatement pstmt = conn
        .prepareStatement("UPDATE "+ dbPrefix+ "consequence SET
objId=?, attrId=?, type=?, value=?, function=?, seqNo=? WHERE id=?");
    try {
      pstmt.setInt(1, c.getObjId());
      pstmt.setInt(2, c.getAttrId());
      pstmt.setInt(3, c.getType());
      pstmt.setString(4, c.getValue());
      pstmt.setString(5, c.getFunction());
      pstmt.setInt(6, c.getSeqNo());
      pstmt.setInt(7, c.getId());
      pstmt.executeUpdate();
    } finally {         pstmt.close();     }  }
  public void deleteConsequence(String[] ids) throws SQLException {
    deleteWithCol(ids, "consequence", "id");   }  /*
   * ###############FORMS################### */
  public void addForm(InputForm i) throws SQLException {
    PreparedStatement pstmt = conn
        .prepareStatement("INSERT INTO "+ dbPrefix
+ "inputform(rulesetId, attrId, question, inputtype, choices, objId, status)
VALUES(?,?,?,?,?,?,?)");
    try {
      pstmt.setInt(1, i.getRulesetId());
      pstmt.setInt(2, i.getAttrId());
      pstmt.setString(3, i.getQuestion());
      pstmt.setInt(4, i.getInputType());
      pstmt.setString(5, i.getChoices());
      pstmt.setInt(6, i.getObjId());
      pstmt.setBoolean(7, i.isStatus());
      pstmt.executeUpdate();
    } finally {         pstmt.close();     }
    formset(i.getRulesetId(), true);   //ruleset's input form is set  }
  public ArrayList<InputForm> getForms(int ruleId, boolean showAll) throws
SQLException {
    ArrayList<InputForm> forms = new ArrayList<InputForm>();
    PreparedStatement pstmt;
    if(showAll)
      pstmt = conn.prepareStatement("SELECT * FROM "
        + dbPrefix + "inputform WHERE rulesetId=?");
    else
      pstmt = conn.prepareStatement("SELECT * FROM "
        + dbPrefix + "inputform WHERE rulesetId=? AND status=?");
    try {
      pstmt.setInt(1, ruleId);
      if(!showAll)            pstmt.setBoolean(2, true);
      ResultSet rs = pstmt.executeQuery();
      try {
        InputForm i;            // get returned id
        while (rs.next()) {
          i = new InputForm();
          i.setRulesetId(ruleId);
          i.setAttrId(rs.getInt("attrId"));
          i.setQuestion(rs.getString("question"));
          i.setInputType(rs.getInt("inputtype"));
          i.setChoices(rs.getString("choices"));
          i.setObjId(rs.getInt("objId"));
          i.setStatus(rs.getBoolean("status"));
          forms.add(i);              }
      } finally {             rs.close();       }
    } finally {         pstmt.close();     }
    for (InputForm i : forms) {
      i.setAttribute(getAttribute(i.getAttrId()));     }
    return forms;   }
  public void updateForm(InputForm i) throws SQLException {
    PreparedStatement pstmt = conn
        .prepareStatement("UPDATE "+ dbPrefix  + "inputform SET status=?,
question=?, inputtype=?, choices=? WHERE rulesetId=? AND attrId=?");
    try {
```

```
            pstmt.setBoolean(1, i.isStatus());
            pstmt.setString(2, i.getQuestion());
            pstmt.setInt(3, i.getInputType());
            pstmt.setString(4, i.getChoices());
            pstmt.setInt(5, i.getRulesetId());
            pstmt.setInt(6, i.getAttrId());
            pstmt.executeUpdate();
        } finally {         pstmt.close();     }  }
    public void deleteForm(int rulesetId, int attribute) throws SQLException {
        PreparedStatement pstmt = conn.prepareStatement("DELETE FROM "
                + dbPrefix + "inputform WHERE rulesetId=? AND attrId=?");
        try {
            pstmt.setInt(1, rulesetId);
            pstmt.setInt(2, attribute);
            pstmt.executeUpdate();
        } finally {         pstmt.close();     }  }
    /* ############GROUPINGS#################  */
    public void addRuleGroup(Rulegroup rg) throws SQLException{
        PreparedStatement pstmt = conn.prepareStatement("INSERT INTO "
    + dbPrefix+ "rulegroup (name, created_by, date_created, description)
VALUES(?,?,?,?)");
        try {
            pstmt.setString(1, rg.getName());
            pstmt.setString(2, rg.getCreated_by());
            pstmt.setString(3, rg.getDate_created());
            pstmt.setString(4, rg.getDescription());
            pstmt.executeUpdate();
        } finally {         pstmt.close();     }  }
    public ArrayList<Rulegroup> getRulegroups() throws SQLException{
        ArrayList<Rulegroup> group = new ArrayList<Rulegroup>();
        PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM "
                + dbPrefix + "rulegroup");
        try {
            ResultSet rs = pstmt.executeQuery();
            try {// insert returned values to the arraylist
                Rulegroup rg;
                while (rs.next()) {
                    rg = new Rulegroup();
                    rg.setId(rs.getInt("id"));
                    rg.setName(rs.getString("name"));
                    rg.setCreated_by(rs.getString("created_by"));
                    rg.setDate_created(rs.getString("date_created"));
                    rg.setDescription(rs.getString("description"));
                    group.add(rg);              }
            } finally {             rs.close();       }
        } finally {         pstmt.close();     }
        return group;   }
    public HashMap<Integer, String> getHashRulegroups() throws
SQLException{
        HashMap<Integer, String> group = new HashMap<Integer, String>();
        PreparedStatement pstmt = conn.prepareStatement("SELECT id,name
FROM "+ dbPrefix + "rulegroup");
        try {
            ResultSet rs = pstmt.executeQuery();
            try {// insert returned values to the arraylist
                while (rs.next()) { group.put(rs.getInt("id"), rs.getString("name")); }
            } finally {             rs.close();       }
        } finally {         pstmt.close();     }
        return group;   }
    public void deleteRulegroups(String[] ids) throws SQLException{
        deleteWithCol(ids, "rulegroup", "id"); }
    /* ############TEMPLATES################ */
    public void addTemplate(Template t) throws SQLException {
        PreparedStatement pstmt = conn.prepareStatement("INSERT INTO "
    + dbPrefix+ "template(name, username, date_uploaded) VALUES(?,?,?)");
        try {
            pstmt.setString(1, t.getName());
            pstmt.setString(2, t.getUsername());
            pstmt.setString(3, t.getDate_uploaded());
            pstmt.executeUpdate();
        } finally {         pstmt.close();     }  }
    public ArrayList<Template> getTemplates() throws SQLException {
        ArrayList<Template> templates = new ArrayList<Template>();
        PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM "
                + dbPrefix + "template");
        try {
            ResultSet rs = pstmt.executeQuery();
            try {
                Template t;     // get returned id
                while (rs.next()) {
                    t = new Template();
                    t.setId(rs.getInt("id"));
                    t.setName(rs.getString("name"));
                    t.setUsername(rs.getString("username"));
                    t.setDate_uploaded(rs.getString("date_uploaded"));
                    templates.add(t);           }
            } finally {             rs.close();       }
        } finally {         pstmt.close();     }
        return templates; }
    public void deleteTemplates(String[] ids) throws SQLException {
        deleteWithCol(ids, "template", "id");   }
    /* ############GENERAL_METHODS_################ */
    private void deleteWithCol(String[] ids, String table, String col) throws
SQLException {
        String query = "DELETE FROM " + dbPrefix + table + " WHERE ";
        for (int i = ids.length - 1; i > 0; i--) {
            if(!ids[i].isEmpty())   query += col + "=" + ids[i] + " || ";       }
        if(!ids[0].isEmpty())  query += col + "=" + ids[0] + "";
        Statement stmt = conn.createStatement();
        try {     stmt.executeUpdate(query);
        } finally {         stmt.close();    }  }
    private int getRecentId(String table) throws SQLException {
        int id = 0;
        PreparedStatement pstmt = conn.prepareStatement("SELECT MAX(id)
FROM "+ dbPrefix + table);
        try {
            ResultSet rs = pstmt.executeQuery();
            try {     if (rs.next()) {       id = rs.getInt("MAX(id)");    }
            } finally {             rs.close();       }
        } finally {         pstmt.close();     }
        return id; }
    public String getIdsWithCol(String[] ids, String table, String col) throws
SQLException{
        String newIds = "";
        String query = "SELECT id FROM " + dbPrefix + table + " WHERE ";
        for (int i = ids.length - 1; i > 0; i--) {
            if(!ids[i].isEmpty())   query += col + "=" + ids[i] + " || ";   }
        if(!ids[0].isEmpty())     query += col + "=" + ids[0] + "";
        Statement stmt = conn.createStatement();
        try {
            ResultSet rs = stmt.executeQuery(query);
            try{  while(rs.next())    newIds += rs.getInt("id") + ",";
            } finally{              rs.close();       }
        } finally {         stmt.close();    }
        return newIds; }
    public HashMap<Integer, String> getTypes(String table) throws
SQLException {
        HashMap<Integer, String> itypes = new HashMap<Integer, String>();
        String query = "SELECT * FROM "+ dbPrefix + table;
        Statement stmt = conn.createStatement();
        try {
            ResultSet rs = stmt.executeQuery(query);
            try { // insert returned values to the arraylist
                while (rs.next()) { itypes.put(rs.getInt("id"), rs.getString("type"));   }
            } finally {             rs.close();       }
        } finally {         stmt.close();    }
        return itypes;   }
    public HashMap<Integer, String> getOperators() throws SQLException {
        HashMap<Integer, String> itypes = new HashMap<Integer, String>();
        PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM "
```

```
            + dbPrefix + "operator");
        try {
            ResultSet rs = pstmt.executeQuery();
            try {// insert returned values to the arraylist
                while (rs.next()) { itypes.put(rs.getInt("id"), rs.getString("operator"));}
            } finally {             rs.close();       }
        } finally {         pstmt.close();     }
        return itypes;   }
    public ArrayList<SelectItem> getSelectOperators() throws SQLException {
        ArrayList<SelectItem> items = new ArrayList<SelectItem>();
        PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM "
            + dbPrefix + "operator");
        try {
            ResultSet rs = pstmt.executeQuery();
            try {// insert returned values to the arraylist
                SelectItem si;
                while (rs.next()) {
                    si = new SelectItem(rs.getInt("id"), rs.getString("name"));
                    items.add(si);              }
            } finally {             rs.close();       }
        } finally {         pstmt.close();     }
        return items;   }  }
```

/WEB-INF/classes/drools/main/ClassManager.java

```
package drools.main;
import general.Config;import general.FileManager;import
general.Globals;import general.Validator;import java.io.IOException;import
java.io.StringWriter;import java.sql.SQLException; import java.util.ArrayList;
import javax.tools.JavaCompiler;import javax.tools.ToolProvider;import
beans.Attribute;
public class ClassManager {
    private final String PATH = Config.getPath();;
    private final String PACKAGE = "drools.classes";
    private StringWriter sw;
    private String className;
    private ArrayList<Attribute> attrs;
    private String var;
    private String var2;
    private String type;
    public ClassManager() { }
    public void init(String name, int objID, ArrayList<Attribute> attrs) throws
SQLException {
        className = name;
        this.attrs = attrs;  }
    public int writeClass() throws IOException, SQLException{
        sw = new StringWriter();
        sw.append("package " + PACKAGE + ";");
        sw.append("\n\npublic class " + className + "{");
        // write constructor
        sw.append("\n\t public " + className + "(){}");
        // for each attribute, declare variable and write getters and setters
        for (Attribute a : attrs) {
            if(!a.isRemoved()){
                var = a.getVariable();
                var2 = Validator.toUpperFirst(var);
                type = Globals.getDataType(a.getDatatype());
                if(type.equals("array"))        type = "String[]";
                declare();
                getter();
                setter();           }       }
        sw.append("\n}");
        FileManager.writeClass(className, sw);  //write file then compile
        return compile(); }
    private void declare() {// declare variable
        sw.append("\n\n\t private " + type + " " + var + ";");   }
    private void getter() {
        sw.append("\n\n\t public " + type + " get" + var2 + "(){");
        sw.append("\n\t\t return " + var + "; }");  }
private void setter() {
        sw.append("\n\n\t public void set" + var2 + "(" + type + " " + var
            + "){");  sw.append("\n\t\t this." + var + "=" + var + "; \n\t}");   }
private int compile() {
        JavaCompiler compiler = ToolProvider.getSystemJavaCompiler();
        String s = PATH + Config.JAVA + className + ".java";
        int result =    compiler.run(null, null, null, s);
        return result;   }
    public void test(String classname){
        className = classname;
        compile();}
    public static void main(String[] args) {
        ClassManager cm = new ClassManager();
        cm.test("test");}  }
```

/WEB-INF/classes/drools/main/DRLCreator.java

```
package drools.main;
import general.FileManager;import general.Globals;import general.Validator;
import java.io.IOException;import java.io.StringWriter;import
java.sql.SQLException;import java.util.ArrayList;import java.util.HashMap;
import beans.Condition;import beans.Consequence;import beans.Function;
import beans.Parameter;import beans.Rule;import beans.Ruleset;
public class DRLCreator {
    private static final String PACKAGE = "drools.rules";
    private static final String IMPORT = "drools.classes.*";
    private static final String GLOBAL = "beans.Result result";
    private static final String AGENDA = "agenda-group";
    private static final String ADDRULE = "result.addRule";
    private static final String ADDDEC = "result.addDecision";
    private static final String ADDOUT = "result.addOutput";
    private static final String SETFOCUS = "drools.setFocus";
    private static final String NOLOOP = "no-loop true";
    private static final String MODIFY = "modify";
    private static final String EXIT = "drools.halt()";
    private static final String VARPREFIX = "var_";
    private static final String EVAL = "eval";
    private static final String FUNCTION = "function";
    private Ruleset ruleset;
    private StringWriter sw;
    //helpers/temps
    private HashMap<String, ArrayList<Parameter>> hashFuncParams;  //<func
name, parameters>
    private ArrayList<Integer> objects;
    private ArrayList<Integer> attributes;
    private String obj;
    private String attr;
    private String params;
    private String conseq;
    private String temp;//for all
    private String conditions;
    private String consequences;
    private String declarations;
    private String explanation;
    public DRLCreator(Ruleset ruleset) {     this.ruleset = ruleset;  }
    public void writeDRL() throws SQLException, IllegalArgumentException,
IOException {
        sw = new StringWriter();
        sw.write(" package " + PACKAGE + ";");
        sw.write("\n import " + IMPORT + ";");
        sw.write("\n global " + GLOBAL + ";");
        initHashFunctions(ruleset.getArrFunctions());   //initialize function details
        writeRules();
        writeFunctions(); //write function
        RuleEngine.compileCheck(sw); }
    public void writeDRLFile(int id) throws IOException{
        FileManager.writeDRL(id, sw);   }
        private void writeRules() throws SQLException{
        for (Rule rule : ruleset.getArrRules()) {
            if(!rule.isRemoved()){
                objects = new ArrayList<Integer>();   //for checking declared objects
and attributes
                attributes = new ArrayList<Integer>();
```

```
            conditions = "";
            consequences = "";
            explanation = "";
            declarations = "";
            //get consequence and conditions
            writeConsequences(rule.getArrCons());
            writeConditions(rule.getArrCond());
            writeDeclarations();
            sw.write("\n\n rule \"" + rule.getRuleId() + "\"");
            sw.write("\n\t salience " + rule.getSalience());
            if(ruleset.getType()==2 && !rule.isStart())//sequential, no need for
agenda groups
              sw.write("\n\t " + AGENDA + " \"" + rule.getAgenda() + "\"");
            if(rule.isNo_loop())  sw.write("\n\t " + NOLOOP);
            sw.write("\n\t when ");
            sw.write(declarations);
            sw.write(conditions);
            sw.write("\n\t then ");
            sw.write("\n\t\t" + ADDRULE + "(\"" + explanation + "\");");
            sw.write(consequences);
            sw.write("\n end ");        }      }  }
   private void writeFunctions() throws SQLException{
      sw.write("\n\n");
      for(Function f: ruleset.getArrFunctions()){
         if(!f.isRemoved()){
            sw.write(FUNCTION + " " + Globals.getDataType(f.getReturntype())
+ " " + f.getName() + "(");
            params = "";
            for(Parameter p: f.getParams()){
               params +=
Globals.getDataType(Globals.getAttrDatatype(p.getAttrId())) + " " +
p.getVariable() + "," ;             }
            if(!params.isEmpty())
               params = params.substring(0, params.lastIndexOf(","));
            sw.write(params + ")");
            sw.write("{ \n\t" + f.getCode() + "\n}");       }      }  }
   private void writeConditions(ArrayList<Condition> con) throws SQLException
{
      for(Condition c : con) {
         if(!c.isRemoved()) {
            conditions += "\n\t\t";
            temp = getBooleanLogic(c.getBool());
            conditions += (temp + " ");
            explanation += (temp + " ");
            attr = Globals.getAttributeName(c.getAttrId());
            obj = Globals.getObjectName(c.getObjId());
            if(c.getType()>1){    //function was used, need EVAL
               conditions += EVAL + "(";
               if(c.isShowObjAttr()){
                  if(!attributes.contains(attr))  //update list of attrs to declare
                     attributes.add(c.getAttrId());
                  conditions += "$"+attr;
                  explanation += obj + "'s " + attr;
               } else{  //showfuncleft
                  temp = writeFunctionCall(c.getFunction());
                  conditions += temp;
                  explanation += temp;                }
               writeOperation(attr, c.getOptrId(), c.getValue(),
c.isShowFuncRight());//if other value is function or custom value
               conditions += ")";// )-close eval      }
            else{//usual condition, obj.attr=value
               conditions += obj + "(";
               explanation += obj + "'s ";
               writeOperation(attr, c.getOptrId(), c.getValue(), false);
               conditions += ")";              }         }      }  }
   private void writeConsequences(ArrayList<Consequence> con) throws
SQLException{
      for(Consequence c: con){
         conseq = Globals.getConseq(c.getType());
         if(c.getObjId()>0)        obj = Globals.getObjectName(c.getObjId());
         if(c.getAttrId()>0)       attr = Globals.getAttributeName(c.getAttrId());
         consequences +=("\n\t\t");
         if(conseq.equals("exit"))             consequences += (EXIT + ";");
         else if(conseq.equals("go to"))
            consequences += (SETFOCUS + "(\"" + c.getValue() + "\");");
         else if (conseq .equals("decision")) { // decision
            consequences += (ADDDEC + "(\"" + c.getValue() + "\"");
            if(c.isShowMethods()) //also output result from function call
               consequences += "+ " + writeFunctionCall(c.getFunction());
            consequences += ");";
         }
         else if (conseq .equals("output value")) { // output value

            if(!objects.contains(c.getObjId()))   //update objects to declare
               objects.add(c.getObjId());
            consequences += (ADDOUT + "(\"" + obj + "\" , \"" + attr + "\" , "
                  + VARPREFIX + obj + ".get"
                  + Validator.toUpperFirst(attr)
                  + "()+\"\" );" ); // appended   +""   to cast values to String   }
         else if (conseq.equals("modify")) { // modify
            if(!objects.contains(c.getObjId()))   //update objects to declare
               objects.add(c.getObjId());
            consequences += (MODIFY + "(" + VARPREFIX + obj + "){\n"
                  + "\t\t\t set"  + Validator.toUpperFirst(attr)+ "(");
            //check if modify value is from inputtext or function
            if(c.isShowMethods()){
               consequences += (writeFunctionCall(c.getFunction()));    }
            else{       consequences += (c.getValue());          }
            consequences +=(")" + "\n\t\t}");       }      }         }
   private void writeDeclarations() throws SQLException{
      int objId;
      for(int i: attributes){
         declarations += "\n\t\t";
         attr = Globals.getAttributeName(i);
         objId = Globals.getAttrObject(i);
         obj = Globals.getObjectName(objId);
         if(objects.contains(objId)){
            declarations += (VARPREFIX + obj + ": ");
            objects.remove((Object)objId);   //remove declared objects         }
         declarations += (obj + "($" + attr + ": " + attr + ")");       }
      for(int i: objects){
         declarations += "\n\t\t";
         obj = Globals.getObjectName(i);
         declarations += (VARPREFIX + obj + ": " + obj + "()");     }      }
   /*   * HELPERS    */
   private void writeOperation(String attr, int optrId, String value, boolean
isFunction) throws SQLException{
      String[] val;
      String optr = Globals.getOperator(optrId);
      if(optr.equalsIgnoreCase("between")){
         val = value.split(",");
         conditions +=(attr + " >= " + val[0] + " && " + attr + " <= " + val[1] );
         explanation += attr + " is between " + val[0] + " and " + val[1] + " ";  }
      else if(optr.equalsIgnoreCase("not between")){
         val = value.split(",");
         conditions +=(attr+ " < " + val[0] + " || " + attr + " > " + val[1] );
         explanation += attr + " is not between " + val[0] + " and " + val[1];   }
      else {
         //write optr and value
         conditions += attr + " " + optr + " ";
         explanation += attr + " " + optr + " ";
         if(isFunction)   temp = writeFunctionCall(value);
         else temp = value;
         conditions += temp ;
         explanation += temp;     }  }
   private String writeFunctionCall(String funcName) throws SQLException{
      String temp = funcName + "(";
      params = "";
      for(Parameter p: hashFuncParams.get(funcName)){
         if(!attributes.contains(p.getAttrId()))    attributes.add(p.getAttrId());
```

```
        params += ("$" + Globals.getAttributeName(p.getAttrId()) + " ,");     }
      if(!params.isEmpty())
        params = params.substring(0, params.lastIndexOf(","));   //remove extra
comma
      temp += params + ")";
      return temp;}
   //condition helpers
   private String getBooleanLogic(int i){
      if(i==0)     return "or ";
      else if(i==1)     return "and ";      else return "";  }
   private void initHashFunctions(ArrayList<Function> functions){
      hashFuncParams = new HashMap<String, ArrayList<Parameter>>();
      for(Function f: functions){
        hashFuncParams.put(f.getName(), f.getParams());      }  }  }
```

/WEB-INF/classes/drools/main/GenericObject.java

```
package drools.main;
import general.Globals;import general.Validator;import
java.lang.reflect.InvocationTargetException;import
java.lang.reflect.Method;import java.sql.SQLException;import
java.util.IllegalFormatException;
public class GenericObject {
   private static final String PACKAGE = "drools.classes.";
   private Object object;
   private Class<?> objClass;
   private Method method;
   public GenericObject(String className) throws ClassNotFoundException,
        InstantiationException, IllegalAccessException, SQLException {
      objClass = Class.forName(PACKAGE + className);
      object = objClass.newInstance();   }
   public void setVar(String var, String value, int dtype)
        throws SecurityException, NoSuchMethodException,
        IllegalArgumentException, IllegalAccessException,
        InvocationTargetException, IllegalFormatException, SQLException {
      String type = Globals.getDataType(dtype);
      if (type.equalsIgnoreCase("int")){
        if(value==null)          value = "0";
        method = objClass.getDeclaredMethod(getMethodName(var),
int.class); method.invoke(object, Integer.parseInt(value));    }
      else if(type.equalsIgnoreCase("String")){
        if(value==null)       value = "";
        method = objClass.getDeclaredMethod(getMethodName(var),
String.class);      method.invoke(object, value);    }
      else if(type.equalsIgnoreCase("double")){
        if(value==null)       value = "0";
        method = objClass.getDeclaredMethod(getMethodName(var),
double.class);     method.invoke(object, Double.parseDouble(value));    }
      else if(type.equalsIgnoreCase("boolean")){
        if(value==null)          value = "false";
        method = objClass.getDeclaredMethod(getMethodName(var),
boolean.class);   method.invoke(object, Boolean.parseBoolean(value));    }
      else if(type.equalsIgnoreCase("char")){
        if(value==null)     value = "";
        method = objClass.getDeclaredMethod(getMethodName(var),
char.class);     method.invoke(object, value.charAt(0));       }
      else if(type.equalsIgnoreCase("array")){
        if(value==null)          value = ",";
        else if(!value.contains(","))          value += ",";
        String [] array =  value.split(",");
        method = objClass.getDeclaredMethod(getMethodName(var),
String[].class);        method.invoke(object, (Object)array);          }  }
   /*  * for checkbox and arrays only   */
   public void setVar2(String var, String[] values)
      throws SecurityException, NoSuchMethodException,
      IllegalArgumentException, IllegalAccessException,
      InvocationTargetException, IllegalFormatException {
      if(values==null)          values = new String[]{""};
      method = objClass.getDeclaredMethod(getMethodName(var),
String[].class);
      method.invoke(object, (Object)values); }
   /** * returns proper method name (set<Name>) for setting values of
variables  * @param var * @return    */
   private String getMethodName(String var) {
      return "set" + Validator.toUpperFirst(var);  }
   public Object getObject() {     return object;  }  }
```

/WEB-INF/classes/drools/main/REManager.java

```
package drools.main;
import java.sql.SQLException;import java.util.ArrayList;import
java.util.HashMap;import beans.Result;import beans.Ruleset;import
beans.InputForm;import database.DbManager;
public class REManager {
   private static ArrayList<Ruleset> publishedRulesets;
   private static HashMap<Integer, RuleEngine> hashRE
   private static HashMap<Integer, String> hashObjName;
   public REManager() { }
   private static void init() throws SQLException {     hashObjName =
DbManager.getInstance().getObjectHash();
      publishedRulesets = DbManager.getInstance().getPublishedRulesets();
      hashRE = new HashMap<Integer, RuleEngine>();
      RuleEngine re;
      for (Ruleset r : publishedRulesets) {
        re = new RuleEngine(r.getId());
        hashRE.put(r.getId(), re);      }        }
   public static ArrayList<Ruleset> getPublishedRulesets() throws
SQLException{
      if(publishedRulesets==null)       init();
      return publishedRulesets;  }
   /** * add ruleset a rule engine instance for the ruleset to hashmap of
published ruleset * @param r Ruleset    * @throws SQLException   */
   public static void publishRuleset(Ruleset r) throws SQLException {
      RuleEngine re = new RuleEngine(r.getId());
      if(hashRE==null || publishedRulesets==null)     init();
      publishedRulesets.add(r);
      hashRE.put(r.getId(), re);   }
   public static void updatePublishedRuleset() throws SQLException{
      publishedRulesets = DbManager.getInstance().getPublishedRulesets();
      hashRE = null;}
   /** * remove ruleset from the hashmap of published ruleset
    * @param r Ruleset    */
   public static void unpublishRuleset(int id) {
      if(hashRE!=null){
        hashRE.remove(id);
        for(Ruleset r: publishedRulesets){
          if(r.getId()==id){
            publishedRulesets.remove(r);
            break;          }       }     }  }
   /** * @param rulesetId id of ruleset to run
    * @param forms contains mapping of data needed by the engine
    * @return Result object  * @throws Exception  */
   public Result runEngine(int rulesetId, ArrayList<InputForm> forms)
        throws Exception {
      if(hashRE==null) hashRE = new HashMap<Integer, RuleEngine>();
      RuleEngine re;
      if(!hashRE.containsKey(rulesetId)){
        re = new RuleEngine(rulesetId);
        hashRE.put(rulesetId, re);     }
      else re = hashRE.get(rulesetId);
      return re.runRules(mapValues(forms, DbManager.getInstance()
          .getObjectsUsed(rulesetId))); }
   /** * map values from the input form to the corresponding objects to input to
the engine   * @param forms * @param objIds    * @return arraylist of objects
with attributes set    * @throws Exception  */
   private ArrayList<Object> mapValues(ArrayList<InputForm> forms,
        String objIds) throws Exception {
      HashMap<Integer, GenericObject> gObj = new HashMap<Integer,
GenericObject>();
      int id;
```

```
        String name;
        for (String s : objIds.split(",")) {
            if (!s.isEmpty()) {     id = Integer.parseInt(s);
                name = hashObjName.get(id);
                gObj.put(id, new GenericObject(name));          }      }
        GenericObject o;
        for (InputForm form : forms) {
            id = form.getAttribute().getObjId();
            o = gObj.get(id);
            if(form.isCheckBox())
                o.setVar2(form.getAttribute().getVariable(), form.getValues());
            else
                o.setVar(form.getAttribute().getVariable(), form.getValue(),
                        form.getAttribute().getDatatype());
            gObj.put(id, o);    }
        ArrayList<Object> objects = new ArrayList<Object>();
        for (Integer i : gObj.keySet()) {objects.add(gObj.get(i).getObject());     }
        return objects; }  }
```

/WEB-INF/classes/drools/main/RuleEngine.java

```
package drools.main;
import general.Config;import java.io.CharArrayReader;import
java.io.StringWriter;import java.util.ArrayList;import java.util.Collection;
import org.drools.KnowledgeBase;import org.drools.KnowledgeBaseFactory;
import org.drools.builder.KnowledgeBuilder;
import org.drools.builder.KnowledgeBuilderError;
import org.drools.builder.KnowledgeBuilderFactory;
import org.drools.builder.ResourceType;
import org.drools.definition.KnowledgePackage;
import org.drools.io.ResourceFactory;
import org.drools.runtime.StatefulKnowledgeSession;import beans.Result;
public class RuleEngine {
    private KnowledgeBase kbase;
    private static final String RESULT = "result";
    public RuleEngine() {  }
    public RuleEngine(int ruleId){     initializeRule(ruleId);   }
    public void initializeRule(int id) {
        String sPath1 = Config.getPath() + Config.DRL + id + ".drl";
        kbase = KnowledgeBaseFactory.newKnowledgeBase();
        KnowledgeBuilder kbuilder = KnowledgeBuilderFactory
                .newKnowledgeBuilder();
        kbuilder.add(ResourceFactory.newFileResource(sPath1),
ResourceType.DRL);
        if (kbuilder.getErrors().size() > 0) {
            String err = "";
            for (KnowledgeBuilderError error : kbuilder.getErrors()) {
                err += error.getMessage() + "\n";
                System.err.println(error);        }
            throw new IllegalArgumentException(err);     }
        Collection<KnowledgePackage> pkgs =
kbuilder.getKnowledgePackages();
        kbase.addKnowledgePackages(pkgs);  }
    public Result runRules(ArrayList<Object> facts) throws Exception {
        StatefulKnowledgeSession ksession =
kbase.newStatefulKnowledgeSession();
        for (Object o : facts) {          ksession.insert(o);      }
        if ((Result) ksession.getGlobal(RESULT) == null) {
            ksession.setGlobal(RESULT, new Result());     }
        ksession.fireAllRules();
        Result d = (Result) ksession.getGlobal(RESULT);
        ksession.dispose();
        interpretResult(d);
        return d;  }
    private void interpretResult(Result d) {
        if (d.getDecisions().isEmpty() && d.getOutput().isEmpty()) {
        } else {        d.print();      }  }
    public static void compileCheck(String name) throws
IllegalArgumentException{
        String sPath1 = Config.getPath() + Config.DRL + "temp_" + name +".drl";
        KnowledgeBuilder kbuilder = KnowledgeBuilderFactory
                .newKnowledgeBuilder();
        // load drl file
        kbuilder.add(ResourceFactory.newFileResource(sPath1),
ResourceType.DRL);
        if (kbuilder.getErrors().size() > 0) {
            String msg = "";
            for (KnowledgeBuilderError error : kbuilder.getErrors()) {
                msg += error.getMessage() + "\n";
                }
            throw new IllegalArgumentException(msg);      }  }
    public static void compileCheck(StringWriter sw) throws
IllegalArgumentException{
        KnowledgeBuilder kbuilder = KnowledgeBuilderFactory
                .newKnowledgeBuilder();
        // load drl file
        kbuilder.add(ResourceFactory.newReaderResource(new
CharArrayReader(sw.toString().toCharArray())), ResourceType.DRL);
        // check for parsing errors
        if (kbuilder.getErrors().size() > 0) {
            String msg = "";
            for (KnowledgeBuilderError error : kbuilder.getErrors()) {
                System.err.println(error);
                msg += error.getMessage() + "\n";        }
            throw new IllegalArgumentException(msg);      }  }  }
```

/WEB-INF/classes/general/Config.java

```
package general;
import java.io.FileNotFoundException;import java.io.FileReader;import
java.io.FileWriter;import java.io.IOException;import java.util.Properties;
import javax.faces.context.FacesContext;import javax.servlet.ServletContext;
/** * Retrieves values from the configuration file (config.properties) under the
 * same package *  * @author gessie *  */
public class Config {
    private static Config config;
    private static Properties properties;
    private static final String SOURCE = "WEB-
INF/classes/general/config.properties";
    private static String path = "";
    private static String context = "";
    //global paths
    public static final String CSS =  "templates/";
    public static final String JAVA = "WEB-INF/classes/drools/classes/";
    public static final String DRL = "WEB-INF/classes/drools/rules/";
    //properties keys
    public static final String DB_URL = "db";
    public static final String DB_USER = "user";
    public static final String DB_PASSWORD = "password";
    public static final String DB_PREFIX = "dbPrefix";
    public static final String SITE_NAME = "site_name";
    public static final String SITE_CSS = "front_template";
    public static final String SITE_URL = "abs_link";
    public static final String SITE_DESC= "site_desc";
    public static final String SITE_MSG = "site_msg";
    public static final String SITE_NO_RESULT = "noresult";
    public static final String BACK_CSS = "back_template";
    /** * Constructor - set to private since this will not be used by other
     * classes. An instance is created only once.  */
    private Config() { }
    /** * Returns a copy of a Config instance to avoid loading the resource
bundle  * over and over   * @throws IOException  */
    public static Config getInstance() {
        if (config == null)         init();
        return config;  }
    private static void init(){
        config = new Config();
        properties = new Properties();
        try {
            Object context =
FacesContext.getCurrentInstance().getExternalContext().getContext();
```

```
        if (context instanceof ServletContext) {
            path = ((ServletContext)context).getRealPath("") + "/";
            context = ((ServletContext)context).getContextPath() + "/";         }
        FileReader fr = new FileReader(path + SOURCE);
        properties.load(fr);
        fr.close();
      } catch (FileNotFoundException e){
        System.out.println("CONFIG FILE NOT FOUND!!!");
      } catch (IOException e){         Messages.err_prop();     }  }
   /** * Returns property values from the config file    */
   public String getValue(String key) {    return properties.getProperty(key);}
   /** * Set new values for the config   */
   public void setValue(String key, String value){
      properties.setProperty(key, value);}
   /** * Re-write the config file for updates  * @throws IOException  */
   public void updateConfig() throws IOException{
      String comment = "Last updated by " + Utils.getCurrentUser()
                  +"\n" + Utils.getCurrentDate();
      FileWriter fw = new FileWriter(path+SOURCE);
      properties.store(fw, comment);
      fw.close();   }
   public static String getPath(){
      if(path==null || path.isEmpty())          init();
      return path; }
   public static String getContext(){
      if(context==null || context.isEmpty())          init();
      return context;}  }
```

/WEB-INF/classes/general/.java

```
package general;
import java.io.BufferedWriter;import java.io.ByteArrayInputStream;import
java.io.CharArrayReader;import java.io.File;import
java.io.FileOutputStream;import java.io.FileReader;import
java.io.FileWriter;import java.io.IOException;import java.io.InputStream;import
java.io.Reader;import java.io.StringWriter;
import java.util.zip.ZipEntry;import java.util.zip.ZipInputStream;
public class FileManager {
   /*  * WRITE FILE */
   public static void writeClass(String name, StringWriter sw)
         throws IOException {
      File file = new File(Config.getPath() + Config.JAVA + name + ".java");
      write(file, new CharArrayReader(sw.toString().toCharArray()));    }
   public static void writeDRLTemp(String name, StringWriter sw)
         throws IOException {
      File file = new File(Config.getPath() + Config.DRL + "temp_" + name +
".drl");
      write(file, new CharArrayReader(sw.toString().toCharArray()));    }
   public static void writeDRL(int id, StringWriter sw)
         throws IOException {
      File file = new File(Config.getPath() + Config.DRL + id + ".drl");
      write(file, new CharArrayReader(sw.toString().toCharArray()));    }
   public static void writeTemplateCss(String filename, FileReader fr) throws
IOException{
      File file = new File(Config.getPath() + Config.CSS + filename + Config.FS
+ filename + ".css");
      write(file, fr);    }
   public static void writeTemplateZip(String filename, byte[] data)
         throws IOException {
      extractZipFile(filename.replace(".zip", ""), new
ByteArrayInputStream(data));}
   private static void write(File file, Reader reader) throws IOException {
      mkdirs(file);
      BufferedWriter writer = null;
      try {
         writer = new BufferedWriter(new FileWriter(file));
         int data = -1;
         while ((data = reader.read()) != -1) {   writer.write(data);       }
      } finally {
         reader.close();
         writer.close();      }  }
   private static void mkdirs(File file) throws IOException {
      if (file.exists() && !file.isFile()) {
         throw new IOException("File " + file.getPath()
               + " is not a file.");    }
      File parentFile = file.getParentFile();
      if (!parentFile.exists() && !parentFile.mkdirs()) {
         throw new IOException("Creating directories "
               + parentFile.getPath() + " failed.");     }  }
   /*   * ZIP FILE   */
   private static void extractZipFile(String name, InputStream is) throws
IOException{
      String destination = Config.getPath() + Config.CSS + name + "/";
      byte[] buf = new byte[1024];
      ZipInputStream zipinputstream = null;
      ZipEntry zipentry;
      zipinputstream = new ZipInputStream(is);
      zipentry = zipinputstream.getNextEntry();
      String newFname;
      while (zipentry != null) {
         newFname = destination + zipentry.getName();
         File newFile = new File(newFname);
         mkdirs(newFile);
         if(newFile.getParent() == null) {
            if(newFile.isDirectory())                  break;           }
         FileOutputStream fileoutputstream = new
FileOutputStream(newFname);
         int n;
         while ((n = zipinputstream.read(buf, 0, 1024)) > -1)
            fileoutputstream.write(buf, 0, n);
         fileoutputstream.close();
         zipinputstream.closeEntry();
         zipentry = zipinputstream.getNextEntry();       }
      zipinputstream.close();   }
   /*   * DELETE FILE   */
   public static void deleteTemplate(String name) throws IOException {
      if (!delete(Config.getPath() + Config.CSS + name))
         Messages.err_delete("template actual file");  }
   public static void deleteObject(String name) throws IOException {
      if (!delete(Config.getPath() + Config.JAVA + name + ".java")
|| !delete(Config.getPath() + Config.JAVA + name + ".class"))
         Messages.err_delete("object actual file");  }
   public static void deleteDRL(String name) throws IOException {
      if (!delete(Config.getPath() + Config.DRL + name + ".drl"))
         Messages.err_delete("DRL ("+name+") actual file"); }
   public static void deleteDRLTemp(String name) throws IOException {
      if (!delete(Config.getPath() + Config.DRL + "temp_" + name + ".drl"))
         Messages.err_delete("DRL ("+name+") actual file"); }
   private static boolean delete(String path) throws IOException {
      File f = new File(path);
      if(f.isDirectory())          return deleteDir(f);
      else return f.delete();  }
   private static boolean deleteDir(File dir){
      if (dir.isDirectory()) {
         String[] children = dir.list();
         for (int i=0; i<children.length; i++) {
            boolean success = deleteDir(new File(dir, children[i]));
         if (!success) { return false; }           }
         return dir.delete();       }
      else return dir.delete();   }  }
```

/WEB-INF/classes/general/Globals.java

```
package general;
import java.sql.SQLException;import java.util.ArrayList;import
java.util.HashMap;import javax.faces.model.SelectItem;
import beans.Attribute;import beans.PageManager;import
database.DbManager;
public class Globals {
   private static PageManager pm = new PageManager();
   private static final String frontTemplate = pm.getFrontTemplate();
   private static String backTemplate = pm.getBackTemplate();
```

```
    private static String siteName = pm.getSiteName();
    public static String getBackTemplate() {    return backTemplate; }
    public static String getSiteName() {       return siteName; }
    public static String getFrontTemplate() {       return frontTemplate;  }
    /*  * HASHMAP & SELECT ITEMS    */
    private static HashMap<Integer, String> dataTypes;
    private static HashMap<Integer, String> inputTypes;
    private static HashMap<Integer, String> operators;
    private static HashMap<Integer, String> conseqTypes;
    private static HashMap<Integer, String> userTypes;
    private static HashMap<Integer, String> objectNames;
    private static HashMap<Integer, Attribute> hashAttribute;
    private static HashMap<Integer, ArrayList<SelectItem>> hashObjAttrs;
    private static HashMap<Integer, String> hashRulegroups;
    private static ArrayList<SelectItem> selectDataTypes;
    private static ArrayList<SelectItem> selectInputTypes;
    private static ArrayList<SelectItem> selectOperators;
    private static ArrayList<SelectItem> selectConseqTypes;
    private static ArrayList<SelectItem> selectUserTypes;
    private static ArrayList<SelectItem> selectObjects;
    private static ArrayList<SelectItem> selectRulegroups;
    //DATA TYPES
    public static String getDataType(int key) throws SQLException{
        if(dataTypes==null)
            dataTypes = DbManager.getInstance().getTypes("datatype");
        return dataTypes.get(key); }
    public static ArrayList<SelectItem> getSelectDataTypes() throws
SQLException{
        if(selectDataTypes==null){
            if(dataTypes==null)
                dataTypes = DbManager.getInstance().getTypes("datatype");
            selectDataTypes = getItems(dataTypes);      }
        return selectDataTypes; }
    //INPUT TYPES
    public static String getInputType(int key)throws SQLException{
        if(inputTypes==null)
            inputTypes = DbManager.getInstance().getTypes("inputtype");
        return inputTypes.get(key);}
    public static ArrayList<SelectItem> getSelectInputTypes() throws
SQLException{
        if(selectInputTypes==null){
            if(inputTypes==null)
                inputTypes = DbManager.getInstance().getTypes("inputtype");
            selectInputTypes = getItems(inputTypes);     }
        return selectInputTypes; }
    // OPERATORS
    public static String getOperator(int key) throws SQLException{
        if(operators==null)
            operators = DbManager.getInstance().getOperators();
        return operators.get(key);   }
    public static ArrayList<SelectItem> getSelectOperator() throws
SQLException{
        if(selectOperators==null){
            selectOperators = DbManager.getInstance().getSelectOperators(); }
        return selectOperators;   }
    // CONSEQUENCE TYPES
    public static String getConseq(int key) throws SQLException{
        if(conseqTypes==null)    conseqTypes =
DbManager.getInstance().getTypes("consequencetype");
        return conseqTypes.get(key);}
    public static ArrayList<SelectItem> getSelectConseqTypes() throws
SQLException{
        if(selectConseqTypes==null){
            if(conseqTypes==null)
                conseqTypes =
DbManager.getInstance().getTypes("consequencetype");
            selectConseqTypes = getItems(conseqTypes);     }
        return selectConseqTypes;}
    // USER TYPES
    public static String getUserType(int key) throws SQLException{
        if(userTypes==null)
            userTypes = DbManager.getInstance().getTypes("usertype");
        return userTypes.get(key); }
    public static ArrayList<SelectItem> getSelectUserTypes() throws
SQLException{
        if(selectUserTypes==null){
            if(userTypes==null)
                userTypes = DbManager.getInstance().getTypes("usertype");
            //selectUserTypes = getItems(userTypes);
            selectUserTypes = new ArrayList<SelectItem>();
            for(Integer i: userTypes.keySet()){
                if(!userTypes.get(i).equals("admin"))
                    selectUserTypes.add(new SelectItem(i, userTypes.get(i)));   }  }
        return selectUserTypes; }
    //OBJECTS AND ATTRIBUTES
    public static String getObjectName(int key) throws SQLException{
        if(objectNames==null)
            objectNames = DbManager.getInstance().getObjectHash();
        return objectNames.get(key); }
    public static ArrayList<SelectItem> getSelectObjects() throws
SQLException{
            objectNames = DbManager.getInstance().getObjectHash();
            selectObjects = getItems(objectNames);
        return selectObjects;  }
    public static String getAttributeName(int key) throws SQLException{
        if(hashAttribute==null)       initHashObjAttrs();
        return hashAttribute.get(key).getVariable();   }
    public static ArrayList<SelectItem> getSelectAttributes(int objId) throws
SQLException{
        if(hashObjAttrs==null)       initHashObjAttrs();
        return hashObjAttrs.get(objId);   }
    public static void initHashObjAttrs() throws SQLException{
        if(objectNames==null)
            objectNames = DbManager.getInstance().getObjectHash();
        hashAttribute = new HashMap<Integer, Attribute>();
        hashObjAttrs = new HashMap<Integer, ArrayList<SelectItem>>();
        ArrayList<SelectItem> temp;
        for(Integer objId: objectNames.keySet()){
            temp = new ArrayList<SelectItem>();
            for(Attribute a: DbManager.getInstance().getAttributes(objId)){
                temp.add(new SelectItem(a.getId(), a.getName()));
                hashAttribute.put(a.getId(), a);          }
            hashObjAttrs.put(objId, temp);      }
        selectObjects = getItems(objectNames);   }
    /** * update changes to list  * @param objId  * @throws SQLException   */
    public static void updateAttrs(int objId) throws SQLException{
        ArrayList<SelectItem> temp = new ArrayList<SelectItem>();
        for(Attribute a: DbManager.getInstance().getAttributes(objId)){
            temp.add(new SelectItem(a.getId(), a.getName()));
            hashAttribute.put(a.getId(), a);       }
        hashObjAttrs.put(objId, temp);   }
    public static int getAttrDatatype(int attr) throws SQLException{
        if(hashAttribute==null)       initHashObjAttrs();
        return hashAttribute.get(attr).getDatatype();  }
    public static int getAttrObject(int attr) throws SQLException{
        if(hashAttribute==null)       initHashObjAttrs();
        return hashAttribute.get(attr).getObjId();    }
    private static ArrayList<SelectItem> getItems(HashMap<Integer, String>
hash){
        ArrayList<SelectItem> arr = new ArrayList<SelectItem>();
        for(Integer i: hash.keySet())       arr.add(new SelectItem(i, hash.get(i)));
        return arr;    }
    public static String getRulegroupName(int id) throws SQLException{
        if(hashRulegroups == null)         initHashRulegroups();
        return hashRulegroups.get(id);   }
    private static void initHashRulegroups() throws SQLException{
        hashRulegroups = DbManager.getInstance().getHashRulegroups();
        selectRulegroups = new ArrayList<SelectItem>();
        for(Integer i: hashRulegroups.keySet())
            selectRulegroups.add(new SelectItem(i, hashRulegroups.get(i)));   }
```

```
   public static ArrayList<SelectItem> getSelectRulegroups() throws
SQLException{
      if(selectRulegroups==null)        initHashRulegroups();
      return selectRulegroups;}
   public static void addRulegroup(int id, String name) throws SQLException{
      if(selectRulegroups==null || hashRulegroups==null)
         initHashRulegroups();
      selectRulegroups.add(new SelectItem(id, name));
      hashRulegroups.put(id, name);  }  }
```

/WEB-INF/classes/general/Messages.java

```
package general;
import java.util.ResourceBundle;import
javax.faces.application.FacesMessage;import
javax.faces.context.FacesContext;
public class Messages {
   private static final String SOURCE = "general/messages";
   private static final ResourceBundle rb =
ResourceBundle.getBundle(SOURCE);
   public static String getMessage(String key){     return rb.getString(key); }
   public static void successCustom(String msg){     setInfoMessage(msg);}
   public static void errorCustom(String msg){      setErrorMessage(msg);  }
   public static String getConfirmDel(String param){
      return "Are you sure you want to delete checked "+ param + "?"; }
   public static String getConfirmRem(String param){
      return "Are you sure you want to remove " + param + "?";}
   public static String getConfirmPub(String param, boolean status){
      if(status)         return "Unpublish ruleset " + param + " ?";
      else return "Publish ruleset " + param + "?";  }
   public static String getConfirmAct(String param, boolean status){
      if(status)         return "Deactivate user account " + param + " ?";
      else return "Activate user account" + param + "?";}
   //======= ERROR MESSAGE
   public static void err_prop(){
      setErrorMessage("Error loading properties file");  }
   public static void err_duplicate(String s){
      setErrorMessage(s + " already exists"); }
   public static void err_update(String s){
      setErrorMessage(s + " update failed."); }
   public static void err_delete(String s){
      setErrorMessage(s + " deletion failed"); }
   public static void err_add(String s){
      setErrorMessage("Error saving new " + s);   }
   public static void err_get(String s){
      setErrorMessage("Error retrieving data for " + s);  }
   public static void err_gets(String s){
      setErrorMessage("Error retrieving " + s);   }
   public static void err_null(String s){
      setErrorMessage(s + " should not be empty");  }
   public static void err_alphaNum(String s){
      setErrorMessage(s + " should be composed of letters and number only");
   }
   public static void err_startName(String s){
      setErrorMessage(s + " should begin with letters only");}
   public static void err_status(String s){
      setErrorMessage("Error in changing " + s + " status");  }
   public static void err_atLeastOne(String s1, String s2){
      setErrorMessage(s1 + " must have at least one (1) " + s2);  }
   public static void err_login(){
      setErrorMessage("Username and password do not match");   }
   public static void err_userDisabled(){
      setErrorMessage("User account disabled."); }
   public static void err_pwdChange(){
      setErrorMessage("Passwords do not match");  }
   public static void err_writeFile(String s){
      setErrorMessage("Error in writing "+ s +" file"); }
   //======== SUCCESS
   public static void suc_save(String s){
      setInfoMessage("New " + s + " saved");}
   public static void suc_update(String s){
      setInfoMessage(s+ " updated"); }
   public static void suc_delete(String s){
      setInfoMessage(s + " deleted.");}
   public static void suc_publish(String s, boolean status){
      if(status)         setInfoMessage(s + " published");
      else setInfoMessage(s + " unpublished");  }
   //========
   protected static void setErrorMessage(String msg) {
      FacesContext.getCurrentInstance().addMessage(null,
            new FacesMessage(FacesMessage.SEVERITY_ERROR, msg,
null));   }
   protected static void setInfoMessage(String msg) {
      FacesContext.getCurrentInstance().addMessage(null,
         new FacesMessage(FacesMessage.SEVERITY_INFO, msg, null));}
   protected static void setWarnMessage(String msg) {
      FacesContext.getCurrentInstance().addMessage(null,
      new FacesMessage(FacesMessage.SEVERITY_WARN, msg, null));}}
```

/WEB-INF/classes/general/PageFilter.java

```
package general;
import java.io.IOException;import javax.servlet.*;import javax.servlet.http.*;
import beans.SessionBean;
public class PageFilter implements Filter {
   private final String RESTRICTED = "/geebbo/admin/restricted.jsf";
   private int type;
   public void destroy() { }
   public void init(FilterConfig filterConfig) throws ServletException {   }
   public void doFilter(ServletRequest request, ServletResponse response,
         FilterChain chain) throws IOException, ServletException {
      HttpServletRequest req = (HttpServletRequest) request;
      HttpServletResponse res = (HttpServletResponse) response;
      String uri = req.getRequestURI();
      SessionBean sb = (SessionBean)
req.getSession().getAttribute("sessionbean");
      if(uri.contains("add2")){
      }else
      if(sb==null || !sb.isLoggedIn()){
         if(!uri.contains("login.jsf")){
            res.sendRedirect("/geebbo/admin/login.jsf");    }      }
      else if(sb!=null && sb.isLoggedIn()){
         type = sb.getType();
         //do not allow to log in if user already logged in
         if(uri.contains("login.jsf"))
            res.sendRedirect("/geebbo/admin/home.jsf");
         else if(uri.contains("/rule/") && type==2)
            res.sendRedirect(RESTRICTED);
         else if(uri.contains("/object/")  && type==1)
            res.sendRedirect(RESTRICTED);
         else if((uri.contains("/user/") || uri.contains("/configuration/")) && type<3)
   res.sendRedirect(RESTRICTED);
         else if(uri.contains("/templates/") || uri.contains("/includes/"));  }
      chain.doFilter(request, response); }  }
```

/WEB-INF/classes/general/Utils.java

```
package general;
import java.util.Calendar;import javax.faces.context.FacesContext;
import beans.SessionBean;
public class Utils {
   public static String getCurrentDate(){
      String date = "";
      Calendar cal = Calendar.getInstance();
      date += cal.get(Calendar.YEAR);
      date += "-" + setLength(cal.get(Calendar.MONTH)+1);
      date += "-" + setLength(cal.get(Calendar.DAY_OF_MONTH));
      date += " " + setLength(cal.get(Calendar.HOUR_OF_DAY));
      date += ":" + setLength(cal.get(Calendar.MINUTE));
      return date; }
   private static String setLength(int i){
      if(i<10)        return "0"+i;
      else        return i+"";   }
```

```
   public static String getCurrentUser(){
      SessionBean sb = (SessionBean)
   FacesContext.getCurrentInstance().getExternalContext().getSessionMap().g
et("sessionbean");
      if(sb!=null)          return sb.getUsername();
      else return "";  }  }
```

/WEB-INF/classes/general/Validator.java

```
package general;
import java.sql.SQLException;
public class Validator {
   /** * A name is considered valid if it is not null and it is composed of letters,
numbers,     * whitespaces and underscores only. The name must begin with
letters only.  * @param name to check   * @param label for the name being
checked   * @return    */
   public static boolean isNameValid(String name, String label) {
      if (name.isEmpty()) {
         Messages.err_null(label);
         return false;    }
      if (!Character.isLetter(name.charAt(0))) { // check beginning
         Messages.err_startName(label);
         return false;    }
      if (hasSpecialChar(name)) { // contains special chars
         Messages.err_alphaNum(label);
         return false;    }
      return true;  }
   /** * Checks if the String contains special characters other than underscore
and whitespaces  * @param s     * @return    */
   private static boolean hasSpecialChar(String s) {
      for (char c : s.toCharArray()) {
         if (!Character.isLetterOrDigit(c) && !Character.isWhitespace(c)
               && !Character.toString(c).equals("_")){     return true;     }     }
      return false; }
   /** *   * @param s     * @return String with first letter in upper case   */
   public static String toUpperFirst(String s){
      char c = s.charAt(0);
      s = Character.toUpperCase(c) + s.substring(1);
      return s;  }
   public static boolean isValueValid(String var, int attrId) throws SQLException
{
      String prefix = "Rule condition error @ input: '"+var + "' - ";
      String name = Globals.getAttributeName(attrId);
      String error = "";
      int dtype;
      if(var==null){error = prefix + name + " must have a value.";     }
      else{
         dtype = Globals.getAttrDatatype(attrId);
         if(dtype==1){   //int
            try{
            Integer.parseInt(var);
            } catch (NumberFormatException e){
               error = prefix + name + " value should be integer.";     }     }
         else if(dtype==2){   //double
            try{
               Double.parseDouble(var);
            } catch (NumberFormatException e){
                  error = prefix + name + " value should be decimal.";       }  }
         else if(dtype==4){   //char
            if(var.length()>1)
               error = prefix + name + " must be composed of a single
character.";
         }
         else if(dtype==5){   //boolean
            if(!"true".equalsIgnoreCase(var) && !"false".equalsIgnoreCase(var))
               error = prefix + name + " can only be 'true' or 'false'.";  }     }
      if(!error.isEmpty()){
         Messages.errorCustom(error);
         return false;
      } else return true;}}
```

X. ACKNOWLEDGEMENT

This paper would not have been made if not for these people with whom I owe my sincerest gratitude for having played important roles, whether big or small, in the accomplishment of this work:

To **Azeus**, for my summer internship, where I first learned about and had a first-hand experience on Expert Systems, Rule Engines and Drools.

To my adviser, **Ms. Carpio**, for the support, understanding and patience.

To all of my **professors** and **lecturers**, especially from **DPSM**, for the different learning experiences we had.

To my sweet **roommates** and **floormates** at CWL, for all your encouragements and patience especially when you, instead of me, are the ones awakened by my alarm.

To **Papa Oh**, for mentioning JSF to me.

To all my blockmates from **Block 12**, for the MP's, departmentals, overnights, birthday and valentine surprises, foods, cards, DDR, (corny) jokes and all the experiences we shared, good or bad... You guys made my UP life so much fun and unforgettable!

To the people who helped me prepare for my defense: **Mara, Yanyan, Sieg** and **Leo**. I can't thank you enough for being there even when I didn't ask you to.

To my "forever friends", **4f++: Mara, Izza, Maita, Joy, Bel** (Need I say the reasons?)

To my **Kevin**, for being there in every step, right from the time this topic first popped into my head up to the time this materialized. Thanks for all the ice cream and sweets you gave me to relieve my stress! Love you!

To my family, to **Daddy** and **Mommy**, for providing for my needs the best way you could and for never failing to make me feel your love, support and belief in me. I dedicate all of these to you two. I love you very much!

And lastly, to **God**, for making all these things possible. *Amen!*